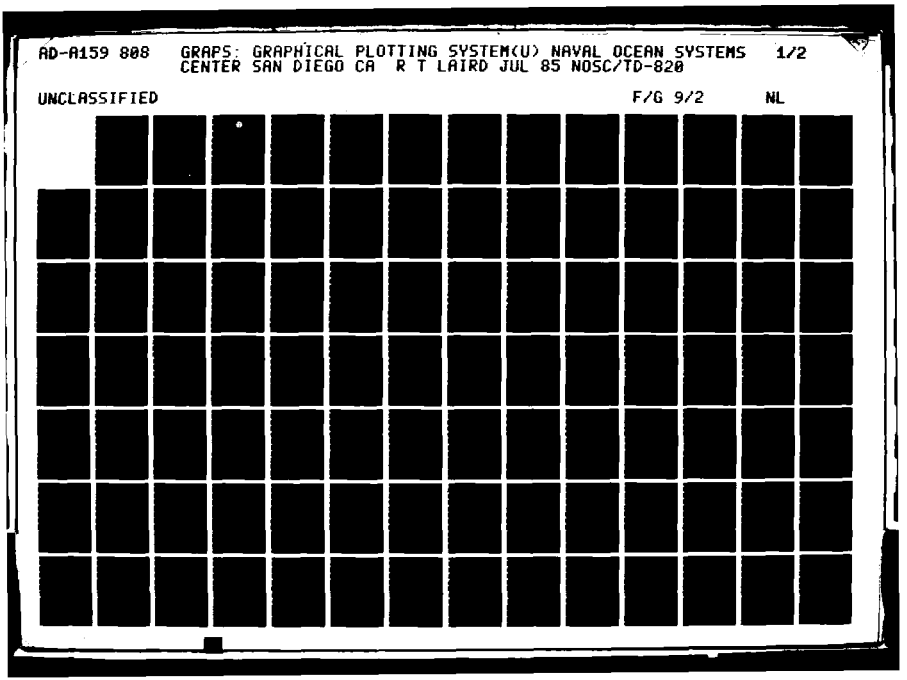
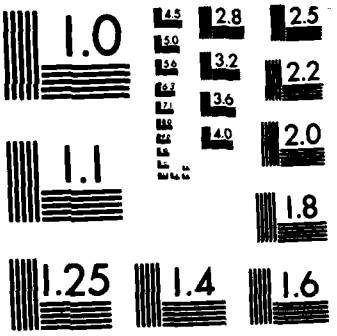

AD-A159 808 GRAPS: GRAPHICAL PLOTTING SYSTEM(U) NAVAL OCEAN SYSTEMS 1/2
CENTER SAN DIEGO CA R T LAIRD JUL 85 NOSC/TD-820
UNCLASSIFIED F/G 9/2 NL



TD 820

(12)

TD 820

Technical Document 820
July 1985

GRAPS: GRAPHICAL PLOTTING SYSTEM

R. T. Laird

Naval Ocean Systems Center
San Diego, CA 92152-5000

NAVAL OCEAN SYSTEMS CENTER SAN DIEGO, CA 92152

AN ACTIVITY OF THE NAVAL MATERIAL COMMAND

F. M. PESTORIUS, CAPT, USN
Commander

R.M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

This work was performed by members of the Antenna and RF Systems Integration Branch, NAVOCEANSYSCEN, for the Naval Electronic Systems Command, Electronic Technology Division, Washington, DC 20363-5100. The task was jointly funded by NELX-614, USACECOM, and USACEEIA.

Released by
I.C. Olson, Head
Antenna and RF Systems
Integration Branch

Under authority of
G.E. Ereckson, Head
Shipboard Systems
Division

RH

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A159 808

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NOSC TD 820	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION Naval Ocean Systems Center 6b. OFFICE SYMBOL (if applicable) Code 822	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Electronic Systems Command 8b. OFFICE SYMBOL (if applicable) NELX-614	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code) Electronic Technology Division Washington, DC 20363-5100	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO.: 62543N PROJECT NO.: XF43454100 TASK NO.: 822-CM41 Agency Accession: DN088 509

11. TITLE (include Security Classification)
GRAPS: GRAphical Plotting System

12. PERSONAL AUTHOR(S)
R. T. Laird

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Final	FROM _____ TO _____	July 1985	150

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	GRAPS: GRAphical Plotting System, BASIC, batch file, control, directory, edit, Epson, function key, graphics, hardware, Integral Data Systems, Leading Edge, Microsoft, modularity, module, MS-DOS, plot

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This document describes operational and design aspects of the GRAphical Plotting System (GRAPS). It can be used as both an introductory guide to the system or as a reference for the experienced user. GRAPS is a program that displays and allows a user to configure and save certain operational parameters of the system. GRAPS also can be used to edit the text labels of graph data files.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL R. T. Laird	22b. TELEPHONE (include Area Code) (619) 225-2646	22c. OFFICE SYMBOL Code 822

DD FORM 1473, 84 JAN

83 APR EDITION MAY BE USED UNTIL EXHAUSTED
ALL OTHER EDITIONS ARE OBSOLETE

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

DD FORM 1473, 84 JAN

CONTENTS

APPENDIX A

APPENDIX B

APPENDIX C

APPENDIX D

1.0 OVERVIEW

This document describes operational and design aspects of the GRAphical Plotting System (GRAPS). It can be used both as an introductory guide to the system and as a reference for the experienced user.

GRAPS is a program that displays and plots graphs of various types produced on a computer. In addition, GRAPS allows a user to configure and save certain operational parameters of the system. GRAPS can also be used to edit the text or label the data of graphs.

This manual covers all aspects of GRAPS (GRAphical Plotting System). It will aid those interested only in the program's operation as well as those concerned with the program's design.

The objective of this project was to implement, on a microcomputer, polar, Smith, linear-linear, linear-log, and log-log charts by using the BASIC language with multiple windows for display of Numerical Electromagnetics Code (NEC) results (ref 1). In addition, a PC-family screen display, matrix printer, and Hewlett-Packard (HP) pen plotter display of the above graph types were to be provided.

The package that I have produced includes all the features outlined above. In addition, facilities for editing text labels within graph data files and for customizing certain operational parameters of the system have been included.

In attempting to follow "good" programming practices, GRAPS has been constructed so that extensions/additions easily can be made. Modularity, on-case oriented control, and software "hooks" have been incorporated into the design of the programs to provide ease of modification. An abundance of documentation, including module and function descriptions, should facilitate system maintenance and understanding.

2.0 PROGRAM DESIGN

The GRAPS program was written by using Microsoft's BASIC language (ref 2). Thus the design of the code is unfortunately limited by the simple constructs of this language. I have, however, tried to maintain the integrity of the programs by using available, and more structured, control sequences such as **WHILE-WEND, IF-THEN-ELSE,** and **ON-GOSUB.** I also tried to avoid the ever unpopular **GOTO,** but found that its use was occasionally "required."[1] The program is written as a stand-alone unit, but could (with a few modifications) be incorporated into a larger system; e.g., the IGUANA system (ref 3).

The GRAPS system consists of several BASIC "modules." Each module is either a subroutine or a collection of logically related program statements that perform a certain task. Most of the modules serve utilitarian roles in carrying out such actions as initializing global/system variables, defining user functions, and maintaining the various screen displays.

GRAPS is a function-driven system. That is, the user is given a menu of the various functions that the system can perform. From this menu, the user selects the functions or sequence of functions that allow him to carry out the desired operation, for example plotting a graph.[2] Thus, in addition to the above utility modules, there are several function modules. Each of these performs a single function that the user can invoke at will. Function-oriented programs allow for a great amount of flexibility in both design and use; modules can be

[1] Use of the **GOTO** statement can, theoretically, always be avoided. However, at times it is just easier (and actually more readable) to use the construct, even at the risk of being kicked out of the computing society.

[2] There are actually three functions involved in plotting a graph: 1) specifying the data file name; 2) specifying the graph plot type; and 3) invoking the *PLOT GRAPH* function.

easily maintained and added to the system, and the user is usually not confined to a rigid sequence of operations that restrict his productivity.[3]

Because it is of a function-oriented nature, GRAPS is driven by a main routine that, after system initialization, waits for commands entered via the function keys, processes these commands, and continues in this manner until the *EXIT* function is chosen. The driver uses a simple looping technique while waiting for commands. I had originally intended to use function key interrupts (**ON-KEY-GOSUB**) to process commands. However, I found that function key interrupts are apparently permanently disabled after an error occurs. As this was obviously a problem, I chose the looping technique.

Another design feature of importance is the array of configuration options available to the user. Certain operating aspects of the system are user-configurable. For example, the user can specify whether a color monitor will be used. Thus, in the design of the various modules, there are certain system variables that must be examined before specific actions can be taken; e.g., you must check the **SYS.COLOR%** variable before using a **COLOR** command. This feature, again, allows for a great amount of flexibility in both programming and use.

3.0 SYSTEM REQUIREMENTS

GRAPS was written with a certain set of hardware/software in mind and, even though this set is not rigidly defined, there are a few particular requirements that are necessary for an operational system.[4]

3.1 HARDWARE

The development system consisted of a Leading Edge PC-XT, a Hewlett-Packard (HP) 7470A plotter, and an Integral Data Systems (IDS) Microprism Model 480 printer. The PC-XT was fitted with 640 kbytes of main memory, a "standard" IBM PC-XT color graphics card, a parallel communications card, and a serial communications card. This system represents the ideal configuration, but any system that is "comparable" should work. These systems should include the following features:

1. **IBM PC-XT compatibility.** That is, the microcomputer must be able to accommodate IBM PC-XT cards and be able to run the software listed below. (If the machine cannot do this, it is not as compatible as you thought!)
2. **256 kbytes of main memory.** This is an approximate lower limit rounded to the nearest 64 k. The actual requirement is probably lower, with the remainder providing a safety margin.
3. **IBM PC-XT-compatible high-resolution graphics card.** This card must be able to support the standard 640 × 200 pixels used in the high-resolution mode of Microsoft's BASIC (see below).
4. **Parallel and serial ports.** These can be obtained in an infinite number of ways by using a variety of interface cards. The ports are used to support the plotter and printer, and thus are not really requirements since the plotter and printer themselves are user options.

---

[3] By allowing the user the freedom to choose which function he/she wises to invoke and when, the repetitious, well-defined routine of using the system is partially removed.

[4] These requirements are specific to GRAPS. Integration of the GRAPS software into another system (e.g., the IGUANA) could possibly introduce additional requirements. I have simply listed those things required to run GRAPS by itself.

5. **HP-compatible plotter.** The plotter must "understand" the HP graphics language (HP-GL; ref 4). Any plotter with this capability will suffice. The following HP plotters understand the HP-GL instruction set: 7470, 7475, 7220, and 9872.
6. **Dot matrix-compatible printer.** There are a number of printers that will work with the GRAPS software. Any printer that is IDS- or Epson-compatible should work. See appendix IV for details on installing different printers.

3.2 SOFTWARE

Software requirements are fewer and less complicated than those for hardware. Any system that meets the hardware requirements and can run Microsoft's MS-DOS version 2.11 (ref 5) and Microsoft BASIC version 2.0 should function properly. All machines that are IBM-compatible will have no problem running this software.

4.0 OPERATION

Before program operation, make sure that the current path includes the directory that contains the Microsoft BASIC software.[5] To use the GRAPS program, issue the following commands:

```
cd graps
basica graps
```

The first command simply changes to the appropriate directory. The second command will load the advanced version of the BASIC interpreter along with the GRAPS program. If the path does not include the BASIC software, the message

```
Bad command or file name
```

will be displayed. Set the path as mentioned above, and then reissue the two commands listed. If the GRAPS software is not in the current directory, the message

```
File not found
Ok
```

will be displayed, and you will be put in the BASIC environment. You must exit the BASIC interpreter and reissue **both** of the commands listed above. Program execution will begin, and the main GRAPS window will be displayed.

Program control proceeds as follows. First, the system variables are initialized. Next, the user functions and function keys are defined. This is followed by the reading of the user-configurable options from the default file *grapsdef.sys*. Finally, the main screen will be displayed. The program will now be awaiting commands from the user.

5.0 THE MAIN SCREEN

The main screen displays the function menu, along with status and error indication lines. In addition, prompt and queue lines are also displayed on the main screen. Most operations (functions) are carried out on this screen. Depending on the configuration options (see below), the screen may or may not appear in color. Also dependent on these options is the display of the status line.

The major component of the main screen is the available function window. This is the box that lists the functions that the user can choose, as well as the corresponding function key that invokes that function. It is used mostly for easy reference.

[5] See the MS-DOS Disk Operating System manual (ref 5), p 103-104, for information on setting the system search path.

The status line appears at the very bottom of the screen and displays information about the current operational state of the system. From left to right, the following fields are displayed: data file name, graph plot type, and configuration file name. Undefined fields are left blank.

When an error occurs (e.g., a bad graph plot type), an error message will appear near the bottom of the screen. The message will be in CAPITAL letters. In addition, if the user has opted to display color, the message will be flashing red letters. The error line remains until the user acknowledges the condition by (usually) pressing a key (see below).

When the user is prompted for input (e.g., when entering a file name), a prompt line is displayed directly below the function window. This line is erased upon a successful entry, but will remain until such an entry is made.

Appearing directly below the error line, and above the status line, is the 'queue' line. This line displays a queue or prompt that informs the user about what is either currently taking place or what to do in response to an error condition. For example, when a plot is being made, the queue line displays the word *plotting* to indicate that the system is currently plotting.

The main screen is replaced, at times, by other screens. The various functions usually alter or replace the main screen altogether. Upon completion of the function, however, the user is returned to the main screen.

6.0 CONFIGURATION OPTIONS

Certain aspects of the program are user-configurable, and are initialized according to the contents of the default configuration file. The default values are as follows:

Color	: FALSE	Date file ext	: DAT
Plotter	: HP7470	Setup file ext	: SYS
Number of Pens	: 2	Status line	: ON
Printer	: IDS	Shell commands	: TRUE

The first thing the user should do is set these parameters according to individual preference. (Changing an option is described below.) The various options and their possible values are explained in the following sections.

6.1 COLOR

The color option determines whether color should be used in the display of the various components of the system. Certain things, such as the status and error lines, are displayed in vibrant colors when this option is set to **TRUE**. When **FALSE**, everything is displayed in black and white.

6.2 PLOTTER

The plotter option actually determines whether a plotter is attached to the system. If there is, the plotter option must be set to one of **HP7470, HP7475, HP7220,** or **HP9872**. If a plotter does not exist, the option should be set to **NONE**.

6.3 NUMBER OF PENS

This option controls the selection of pens when plotting. It should be set to a value between 1 and 8, according to the number of pens the plotter can hold at any one time. As plots are made, GRAPS will cycle through pen *one* to pen *number-of-pens,* using a different pen for each plot. Intelligent use of this option can give a user great control over which pens are used for a plot.

6.4 PRINTER

The printer option is used in a manner similar to the plotter option. It should be set to **IDS** or **EPSON**, according to the compatibility of the selected printer. **NONE** should be entered if there is no printer connected to the system.

6.5 DATA FILE EXTENSION

This option indicates the default data file name extension.[6] It defaults to **.DAT**, and can be set to any three-character, legal file name extension. When entering data file names, for example, or when using the *SET FILE NAME* function, the user has only to input the first part of the file name; the extension will automatically be supplied as the current default value. See the *Data Files* section for more information.

6.6 SETUP FILE EXTENSION

Similarly, this option controls the default setup file name extension. It originally defaults to **.SYS**, and can be set to any three-character, legal file name extension.

6.7 STATUS LINE

This option controls display of the status line that appears at the bottom of the main screen. When **TRUE**, the line is displayed; when **FALSE**, it is not.

6.8 SHELL COMMANDS

This option controls the issuance of shell commands. Shell commands are commands to the operating system issued from the BASIC environment. They require a certain amount of additional memory to execute, and thus are not always practical because of possible constraints upon available memory. A value of **TRUE** allows these commands to pass to the operating system, whereas a value of **FALSE** traps the commands before they reach the OS.

7.0 FUNCTIONS

Once the program has been loaded correctly and the main function menu has been displayed, the user is able to enter commands. Functions are invoked by pressing the function key (**F1 - F10**) corresponding to the desired action. Data entry is made by entering the information and then pressing the<**RETURN**> key. In most instances, if you invoke a function that requires input **and** you do not want to input a new value, typing **RETURN** by itself will produce an exit without changing any old value. The following are descriptions of the functions and how they were intended to be used with GRAPS.

7.1 SET FILE NAME

This function simply allows the user to input the name of the desired data file. The name must be a legal, MS-DOS file name, and should not contain a path specifier.[7] Simply input the name of the file that contains the graph data. Note that a lone<**RETURN**>will abort the function. If an illegal name is given, or if the file cannot be found, an error message is displayed.

---

[6] All data file names must include an extension. Strange and unpredictable things will happen if one does not exist.

[7] This is only a request and not a requirement. If the name includes a path specifier, chances are that the entire path/file name will not fit properly on the status line, resulting in an unattractive appearance.

7.2 SET GRAPH TYPE

To plot a graph, it is necessary to know the type of graph represented by the data. This function allows the user to specify the graph or plot type. Currently, the available types are *linear, bilinear, loglinearh, loglinearv, polar, Smith,* and *loglog.*[8] Enter the plot type that corresponds to the data type in the file chosen above. Note that the entry must be in lower case. If the type chosen is not one of the aforementioned, an error message will be displayed.

7.3 PLOT GRAPH

The plot graph function plots (on the HP7470A plotter) the graph specified by the data within the chosen file. After invoking the function, you will be prompted to ready the plotter (put in the pens, etc.) and then type any key. A message will be displayed at the bottom of the screen indicating that the plot function is activated. Make sure that the plot type and the data within the chosen file are of the same type; that is, do not try to plot a linear graph with polar data.

7.4 VIEW GRAPH

This function allows the user to view (on the Leading Edge PC's screen) the graph specified by the data within the chosen file. The screen will be cleared and the graph will be displayed. When you are finished looking at the graph, type function key **F10**. You will be returned to the main menu.

7.5 PRINT GRAPH

The print graph function prints (on the IDS graphics printer) the graph specified by the data in the chosen file. After invoking the function, you will be prompted to ready the printer (adjust the paper, etc.) and then type any key. The screen will be cleared and the graph will be displayed. Then the image on the screen will be 'dumped' to the printer. Note that this process takes approximately 10 minutes to complete, and should be used only in cases of hardship; e.g., no plotter available.[9] When the operation is finished, the main screen will be displayed. See figures 8 through 14 at the back of the manual for sample screen printouts.

7.6 EDIT LABELS

The edit labels function allows a user a limited editing facility to move, insert, and delete text labels appearing on a graph. The function first displays the graph specified by the data of the chosen file, and then puts the user into an edit mode. A cursor is displayed in the upper left corner of the screen, and can be positioned as desired by using the directional arrows on the numeric keypad.

[8] The names of the graph types describe what is being plotted. For example, bilinear means that both the X and Y axes will be linearly scaled, as opposed to logarithmically scaled. See the subroutine listings and figures 1 through 7 at the back of the manual for details on the plot produced for each of the graph types.

[9] Obtaining a screen dump is actually quite tricky. The operation requires that a screen dump program be loaded and resident before the GRAPS software is used. The print screen function simply generates an interrupt (via an assembly language program), which, in turn, causes a print screen call.

```
12260 'Function to create a filename with the extension E$, i.e., F$.E$
12265     DEF FNFILENAME$(F$, E$) = LEFT$(F$,INSTR(F$,".")) + E$
12270 '
12275 'Function which creates a "centered version" of the parameter string.
12280     DEF FNSTRCNTR$(S$, F) = SPACE$((F - LEN(S$))  2 - 1) + S$
12285 '
12290 'Function to delete the leading space from a (numeric) string.
12295     DEF FNLSD$(S$) = RIGHT$(S$, LEN(S$)-SGN(SGN(VAL(S$))+1))
12300 '
12305 'Function to initialize plotter; sets input points one and two.
12310 'pen number one.
12315     DEF FNINIT.PLOT$(X1, Y1, X2, Y2) = " IN;IP" + STR$(X1)+","+STR$(Y1)+","
                                + STR$(X2)+","+STR$(Y2)+";"
12320 '
12325 'Function to set starting point for line drawing (PEN GOES DOWN).
12330     DEF FNSTART.FROMD$(X, Y) = " PUPA" + STR$(X) + "," + STR$(Y) + "; PD;"
12335 '
12340 'Function to set starting point for line drawing (PEN STAYS UP).
12345     DEF FNSTART.FROMU$(X, Y) = " PUPA" + STR$(X) + "," + STR$(Y)
12350 '
12355 'Function to draw from current point to new point.
12360     DEF FNDRAW.TO$(X, Y) = " PA" + STR$(X) + "," + STR$(Y) + ";"
12365 '
12370 'Function to set direction in which character sare lettered.
12375     DEF FNCHAR.DIR$(DX, DY) = " DI" + STR$(DX) + "," + STR$(DY) + ";"
12380 '
12385 'Function to select character size.
12390     DEF FNCHAR.SIZE$(W, H) = " SI" + STR$(W) + "," + STR$(H) + ";"
12395 '
12400 'Function to print (ready for print) a label.
12405     DEF FNLABEL$(L$) = " LB"+L$+EOT$
12410 '
12415 'Function to select pen color.
12420     DEF FNPEN.COLOR$(C) = " SP" + STR$(C) + ";"
12425 '
12430 'Function to select line type.
12435     DEF FNLINE.TYPE$(T) = " LT" + STR$(T) + ";"
12440 '
12445 'Function to locate pen (at screen-oriented coordinates; horizontally).
12450     DEF FNPHLOCATE$(R, C) = " PUPA"+STR$((C-1)*125)+","+STR$((26-R)*300)
12455 '
12460 'Function to locate pen (at screen-oriented coordinates; vertically).
12465     DEF FNPVLOCATE$(R, C) = " PUPA"+STR$(9500-R*364)+","+STR$(7774-C*124)
12470 '
12475 'Funct on to draw a circle of radius, R, at current pen location.
12480     DEF FNPCIRCLE$(R) = " CI" + STR$(R) + ";"
12485 '
12490 'Function to draw an arc (absolute) to the given X,Y coordinates.
12495     DEF FNPARC$(X, Y, R) = " AA" + STR$(X) +","+ STR$(Y) +","+ STR$(R) +";"
12500 '
12505 'Function to check for validity of graph type.
12510     DEF FNPLOT.TYPE.OK(P$) = (P$ = "LINEAR")    OR (P$ = "BILINEAR")   OR
                  (P$ = "LOGLINEARH") OR (P$ = "LOGLINEARV") OR
                  (P$ = "POLAR")     OR (P$ = "SMITH")     OR
```

FN MODULE

```
12000 REM
12005 REM ************************************************************************
12010 REM *
12015 REM *  Function Name    :     fn
12020 REM *
12025 REM *  Description      :     This file contains various conversion,
12030 REM *                     string comparison, string generation,
12035 REM *                     and other, miscellaneous functions.
12040 REM *
12045 REM *  To Call          :     FNcommand_name(args)
12050 REM *
12055 REM *                     args - The function arguments to the
12060 REM *                          various functions. See the
12065 REM *                          individual functions for the
12070 REM *                          number and type of arguments.
12075 REM *
12080 REM *  Returns          :     Values which are the parameter values
12085 REM *                     processed in some manner, e.g., scaled,
12090 REM *                     concatenated, etc.
12095 REM *
12100 REM *  Edit History     :     1) Robin Laird 3/7/85
12105 REM *
12110 REM ************************************************************************
12115 REM
12120 '
12125 'Function to convert a data value X to plotter coordinates (rectangular).
12130     DEF FNCVTX(X) = (PLOT.X.MAX-PLOT.X.MIN) * X + PLOT.X.MIN
12135 '
12140 'Function to convert a data value Y to plotter coordinates (rectangular).
12145     DEF FNCVTY(Y) = (PLOT.Y.MAX-PLOT.Y.MIN) * Y + PLOT.Y.MIN
12150 '
12155 'Function to convert a data value X to plotter coordinates (polar).
12160     DEF FNCVTXP(X) = HP.Y.MAX * (X+1)/2
12165 '
12170 'Function to convert a data value Y to plotter coordinates (polar).
12175     DEF FNCVTYP(Y) = HP.Y.MAX * (Y+1)/2
12180 '
12185 'Function to convert a data value X to plotter coordinates (smith).
12190     DEF FNCVTXS(K, I) = HP.Y.MAX/2 * (2-(((K-1)*(K+1)+I^2)/((K+1)^2+I^2)+1))
12195 '
12200 'Function to convert a data value Y to plotter coordinates (smith).
12205     DEF FNCVTYS(K, I) = HP.Y.MAX/2 * (2 - ((2*I)/((K+1)^2+I^2)+1))
12210 '
12215 'Function to convert a data value X to screen coordinates (smith).
12220     DEF FNCVTXSS(K, I) = -.15*((K-1)*(K+1)+I^2)+.05
12225 '
12230 'Function to convert a data value Y to screen coordinates (smith).
12235     DEF FNCVTYSS(K, I) = .15*(2*I)
12240 '
12245 'Function to take log of number (base 10).
12250     DEF FNLOG10(X) = LOG(X) / LOG(10)
12255 '
```

```
11800     FOR I = 0 TO 8 : READ SPLUS%(I)  : NEXT I
11805     FOR I = 0 TO 8 : READ SPOINT%(I) : NEXT I
11810 '
11815 'Various plotter default commands.
11820     PEN.COLOR.DEF$ = "SP1;"
11825     LINE.TYPE.DEF$ = "LT;"
11830 '
11835 'Label editing variables including editing cursor.
11840     MAX.LABELS% = 25
11845     DIM LABEL.POS(2, MAX.LABELS%)
11850     DIM LABEL.STR$(MAX.LABELS%)
11855     DIM CSR%(8)
11860 '
11865 'Plotter x and y min/max (dimensions within graph border).
11870     PLOT.X.MIN = 1000     : PLOT.X.MAX = 9000
11875     PLOT.Y.MIN = 960 : PLOT.Y.MAX = 7250
11880 '
11885 'Plotter x and y min/max (absolute, see pg. 2-10 of "HP 7470A Interfacing
11890 'And Programming Manual" for details).
11895     HP.X.MIN = 0      : HP.X.MAX = 10300
11900     HP.Y.MIN = 0      : HP.Y.MAX = 7650
11905 '
11910 'LEPC hi resolution screen dimensions.
11915     HIRES.X.MIN = 0   : HIRES.X.MAX = 639
11920     HIRES.Y.MIN = 0   : HIRES.Y.MAX = 199
11925 '
11930 'Home screen's viewport x and y min/max.
11935     HSCRN.X.MIN = 14: HSCRN.X.MAX = 67
11940     HSCRN.Y.MIN = 4  : HSCRN.Y.MAX = 10
11945 '
11950 'All finished, so return to caller.
11955     RETURN
```

```
11530     SYS.QUE.LOCX = 1      : SYS.QUE.LOCY = 24
11535     SYS.STAT.LOCX = 1     : SYS.STAT.LOCY = 25
11540 '
11545 'Graph (plot) type variables.
11550     NUM.GRAPH.TYPES% = 7
11555     DIM GRAPH.TYPE$(NUM.GRAPH.TYPES%)
11560     FOR I = 1 TO NUM.GRAPH.TYPES% : READ GRAPH.TYPE$(I) : NEXT I
11565 '
11570 'Function key definitions.
11575     NUM.FKEYS% = 10
11580     DIM FKEY$(NUM.FKEYS%)
11585     FOR I = 1 TO NUM.FKEYS%       : READ FKEY$(I)     : NEXT I
11590 '
11595 'Function menu lines.
11600     NUM.MENU.LINES% = 5
11605     DIM MENU.LINE$(NUM.MENU.LINES%)
11610     FOR I = 1 TO NUM.MENU.LINES%  : READ MENU.LINE$(I)  : NEXT I
11615 '
11620 'Print screen routine.
11625     NUM.WORDS% = 2
11630     DIM PRTSC%(NUM.WORDS%)
11635     FOR I = 0 TO NUM.WORDS%-1     : READ PRTSC%(I)      : NEXT I
11640 '
11645 'Graphics characters (A0% through APOINT%).
11650     DIM A0%(8),A1%(8),A2%(8),A3%(8),A4%(8)
11655     DIM A5%(8),A6%(8),A7%(8),A8%(8),A9%(8)
11660     DIM AMINUS%(8), APLUS%(8), APOINT%(8)
11665     DIM S0%(8),S1%(8),S2%(8),S3%(8),S4%(8)
11670     DIM S5%(8),S6%(8),S7%(8),S8%(8),S9%(8)
11675     DIM SMINUS%(8), SPLUS%(8), SPOINT%(8)
11680     FOR I = 0 TO 8 : READ A0%(I) : NEXT I
11685     FOR I = 0 TO 8 : READ A1%(I) : NEXT I
11690     FOR I = 0 TO 8 : READ A2%(I) : NEXT I
11695     FOR I = 0 TO 8 : READ A3%(I) : NEXT I
11700     FOR I = 0 TO 8 : READ A4%(I) : NEXT I
11705     FOR I = 0 TO 8 : READ A5%(I) : NEXT I
11710     FOR I = 0 TO 8 : READ A6%(I) : NEXT I
11715     FOR I = 0 TO 8 : READ A7%(I) : NEXT I
11720     FOR I = 0 TO 8 : READ A8%(I) : NEXT I
11725     FOR I = 0 TO 8 : READ A9%(I) : NEXT I
11730     FOR I = 0 TO 8 : READ AMINUS%(I) : NEXT I
11735     FOR I = 0 TO 8 : READ APLUS%(I)  : NEXT I
11740     FOR I = 0 TO 8 : READ APOINT%(I) : NEXT I
11745     FOR I = 0 TO 8 : READ S0%(I) : NEXT I
11750     FOR I = 0 TO 8 : READ S1%(I) : NEXT I
11755     FOR I = 0 TO 8 : READ S2%(I) : NEXT I
11760     FOR I = 0 TO 8 : READ S3%(I) : NEXT I
11765     FOR I = 0 TO 8 : READ S4%(I) : NEXT I
11770     FOR I = 0 TO 8 : READ S5%(I) : NEXT I
11775     FOR I = 0 TO 8 : READ S6%(I) : NEXT I
11780     FOR I = 0 TO 8 : READ S7%(I) : NEXT I
11785     FOR I = 0 TO 8 : READ S8%(I) : NEXT I
11790     FOR I = 0 TO 8 : READ S9%(I) : NEXT I
11795     FOR I = 0 TO 8 : READ SMINUS%(I) : NEXT I
```

```
11260     DATA 8, 8, 0,    0,   12288, 48,  0, 0, 0
11265 '
11270     DATA 8, 8, 0, -4096, -28528, -3952,  0, 0, 0
11275     DATA 8, 8, 0, 16384, 16576,  -8128,  0, 0, 0
11280     DATA 8, 8, 0, -4096, -4080,  -3968,  0, 0, 0
11285     DATA 8, 8, 0, -4096, -4080,  -4080,  0, 0, 0
11290     DATA 8, 8, 0, 12288, -1968,  4112,   0, 0, 0
11295     DATA 8, 8, 0, -4096, -3968,  -4080,  0, 0, 0
11300     DATA 8, 8, 0, -4096, -3968,  -3952,  0, 0, 0
11305     DATA 8, 8, 0, -4096, 8208,   16448,  0, 0, 0
11310     DATA 8, 8, 0, -4096, -3952,  -3952,  0, 0, 0
11315     DATA 8, 8, 0, -4096, -3952,  -4080,  0, 0, 0
11320     DATA 8, 8, 0, 0,     -8192,  0,      0, 0, 0
11325     DATA 8, 8, 0, 0,     -8128,  64,     0, 0, 0
11330     DATA 8, 8, 0, 0,     0,      -32768, 0, 0, 0
11335 '
11340 'System literals.
11345     FALSE% = 0        : TRUE% = NOT FALSE% : PI = 3.1415926#
11350 '
11355 'Character constants, including control characters.
11360     EOT$ = CHR$(3)   : ESC$ = CHR$(27): BELL$ = CHR$(7)
11365     NORMAL% = 0      : INTENSE% = 8          : BLINK% = 16
11370     CTRL.D$ = CHR$(4)     : CTRL.L$ = CHR$(12)   : CTRL.T$ = CHR$(20)
11375     UPARROW$ = CHR$(0)+"H" : DNARROW$ = CHR$(0)+"P"
11380     LFARROW$ = CHR$(0)+"K" : RTARROW$ = CHR$(0)+"M"
11385 '
11390 'System "state" variables.
11395     SYS.DO.ERRORS% = TRUE% : SYS.ERROR% = FALSE%      : SYS.EXIT% = FALSE%
11400 '
11405 'System parameters (user configurable options).
11410     SYS.COLOR% = FALSE%
11415     SYS.PLOTTER$ = "NONE"
11420     SYS.NUM.PENS% = 2
11425     SYS.PRINTER$ = "NONE"
11430     SYS.DATA.EXT$ = "DAT"
11435     SYS.SETUP.EXT$ = "SYS"
11440     SYS.STAT.LINE% = TRUE%
11445     SYS.SHELL% = FALSE%
11450     SYS.NUM.LINE.TYPES% = 7
11455 '
11460     SYS.FILENAME$ = ""
11465     SYS.GRAPHTYPE$ = ""
11470     SYS.SETUP$ = "GRAPSDEF.SYS"
11475 '
11480     SYS.X.PART = 10
11485     SYS.XL.LABEL = 400    : SYS.XR.LABEL = 9100
11490     SYS.Y.PART = 5
11495     SYS.YB.LABEL = 600    : SYS.YT.LABEL = 7500
11500     SYS.XP.LABEL = 3700   : SYS.YP.LABEL = 4175
11505     SYS.FLAG.VALUE = -1.234
11510 '
11515 'System window locations.
11520     SYS.PROMPT.LOCX = 14      : SYS.PROMPT.LOCY = 12
11525     SYS.ERR.LOCX = 1: SYS.ERR.LOCY = 22
```

INIT MODULE

```
11000 REM
11005 REM ************************************************************************
11010 REM *
11015 REM *  Routine Name:     init
11020 REM *
11025 REM *  Description   :     This subroutine initializes (declares
11030 REM *                   and gives initial values to) all data
11035 REM *                   structures. It should be called before
11040 REM *                   any other routine.
11045 REM *
11050 REM *  To Call       :     GOSUB 11000
11055 REM *
11060 REM *  Globals       :     See below; all variables are affected.
11065 REM *
11070 REM *  Edit History  :     1) Robin Laird 3/7/85
11075 REM *
11080 REM ************************************************************************
11085 REM
11090 '
11095 'Reset data pointer to first item.
11100     RESTORE
11105 '
11110 'Data for graph types.
11115     DATA "LINEAR", "BILINEAR", "LOGLINEARV", "LOGLINEARH"
11120     DATA "POLAR", "SMITH",   "LOGLOG"
11125 '
11130 'Data for function key definitions.
11135     DATA "1", "2", "3", "4", "5", "6", "7", "8", "9", "A"
11140 '
11145 'Data for function menu.
11150     DATA "F1 -> set file name        F2 -> set graph type"
11155     DATA "F3 -> view graph           F4 -> plot graph"
11160     DATA "F5 -> print graph          F6 -> edit labels"
11165     DATA "F7 -> read setup           F8 -> save setup"
11170     DATA "F9 -> change setup          F0 -> exit program"
11175 '
11180 'Data for print screen interrupt (assembly language) routine.
11185     DATA &h05CD, &h00CB
11190 '
11195 'Data for graphics characters ("0123456789-+.", normal and scaled).
11200     DATA  8, 8, 16956, 19014, 25170, 60,  0, 0, 0
11205     DATA  8, 8, 12304, 4176,  4112,  16,  0, 0, 0
11210     DATA  8, 8, 16956, 3074,  16432, 126, 0, 0, 0
11215     DATA  8, 8, 16956, 7170,  16898, 60,  0, 0, 0
11220     DATA  8, 8, 5132,  17444, 1150,  4,   0, 0, 0
11225     DATA  8, 8, 16510, 636,   16898, 60,  0, 0, 0
11230     DATA  8, 8, 16956, 31808, 16962, 60,  0, 0, 0
11235     DATA  8, 8, 638,   2052,  4104,  16,  0, 0, 0
11240     DATA  8, 8, 16956, 15426, 16962, 60,  0, 0, 0
11245     DATA  8, 8, 16956, 15938, 16898, 60,  0, 0, 0
11250     DATA  8, 8, 0,     -512,  0,     0,   0, 0, 0
11255     DATA  8, 8, 4112,  -496,  4112,  16,  0, 0, 0
```

```
10250 '       34000  Error message routine.
10255 '       35000  Graph label retrieval/storage routine.
10260 '       36000  Character (numeric) display routine.
10265 '       .....
10280 '       40000  Linear plotting routine.
10285 '       41000  Bilinear plotting routine.
10290 '       42000  Loglinear (vertical) plotting routine.
10295 '       43000  Loglinear (horizontal) plotting routine.
10300 '       44000  Polar plotting routine.
10305 '       45000  Smith plotting routine.
10310 '       46000  Loglog plotting routine.
10315 '       50000  Linear display routine.
10320 '       51000  Bilinear display routine.
10325 '       52000  Loglinear (vertical) display routine.
10330 '       53000  Loglinear (horizontal) display routine.
10335 '       54000  Polar display routine.
10340 '       55000  Smith display routine.
10345 '       56000  Loglog display routine.
10350 '
10355 'Define functions, initialize data structures, set up function keys,
10360 'and display function menu (main) screen.
10365     ON ERROR GOTO 33000
10370     GOSUB 11000 'call init()
10375     GOSUB 12000 'call fn()
10380     GOSUB 13000 'call defsetup()
10385     GOSUB 14000 'call fkeys()
10390     GOSUB 15000 'call menu()
10395 '
10400 'Loop, inputting key commands. Make sure we always return to menu.
10405     WHILE (NOT SYS.EXIT%)
10410       LOCATE 1,1,1
10415       SYS.SCRN.CHANGED% = FALSE%
10420       G$ = ""
10425       WHILE (G$ = "") : G$ = INKEY$ : WEND
10430       IF G$ = FKEY$(1)  THEN GOSUB 20000 'call getfn
10435       IF G$ = FKEY$(2)  THEN GOSUB 21000 'call getgt
10440       IF G$ = FKEY$(3)  THEN GOSUB 22000 'call view
10445       IF G$ = FKEY$(4)  THEN GOSUB 23000 'call plot
10450       IF G$ = FKEY$(5)  THEN GOSUB 24000 'call print
10455       IF G$ = FKEY$(6)  THEN GOSUB 25000 'call edit
10460       IF G$ = FKEY$(7)  THEN GOSUB 26000 'call rdsetup
10465       IF G$ = FKEY$(8)  THEN GOSUB 27000 'call svsetup
10470       IF G$ = FKEY$(9)  THEN GOSUB 28000 'call chsetup
10475       IF G$ = FKEY$(10) THEN GOSUB 29000 'call exit
10480       IF G$ = CTRL.D$   THEN GOSUB 30000 'call dir
10485       IF G$ = CTRL.L$   THEN GOSUB 31000 'call chdir
10490       IF G$ = CTRL.T$   THEN GOSUB 32000 'call type
10495 '
10500 'Make sure we are displaying main screen.
10505       IF SYS.SCRN.CHANGED% THEN GOSUB 15000 'call menu()
10510     WEND
10515     SYSTEM
```

MAIN MODULE

```
10000 REM ************************************************************************
10005 REM *
10010 REM *  Program Name    :     main (version 1.1)
10015 REM *
10020 REM *  Description     :     This program allows a user to plot
10025 REM *                        or print graphs of several different
10030 REM *                        types. In addition, there are facili-
10035 REM *                        ties for editing the text labels of
10040 REM *                        graphs, as well as a "system customi-
10045 REM *                        zation" facility. See the "GRAPS
10050 REM *                        User's Guide" for a more detailed
10055 REM *                        description of the features and
10060 REM *                        functions of the program.
10065 REM *
10070 REM *  Usage           :     basica graps
10075 REM *
10080 REM *  Edit History    :     1) Robin Laird 3/5/85
10085 REM *
10090 REM ************************************************************************
10095 REM
10100 '
10105 'Below is a map of the various segments of code, e.g., functions and
10110 'subroutines. It shows where things begin. Since BASIC is interprative,
10115 'the placement of code is critical. This leads to a strange and incon-
10120 'venient arrangement of the components of the program.
10125 '
10130 'Code Line Number Map (very subject to change):
10135 '
10140 '       10000 Main program.
10145 '       11000 Data structure initialization.
10150 '       12000 Function definitions.
10155 '       13000 Default set up routine.
10160 '       14000 Function key set up.
10165 '       15000 Display menu screen.
10170 '       16000 Clear screen.
10175 '       17000 Draw window.
10180 '       18000 User queuing routine.
10185 '       19000 System "shell" command routine.
10190 '       20000 Open file function.
10195 '       21000 Set plot type function.
10200 '       22000 View graph function.
10205 '       23000 Plot graph function.
10210 '       24000 Print graph function.
10215 '       25000 Edit labels function.
10220 '       26000 Read system setup.
10225 '       27000 Save system setup.
10230 '       28000 Change system setup.
10235 '       29000 Exit program function.
10240 '       30000 Directory listing      (^D).
10245 '       31000 Change logged directory (^L).
10250 '       32000 Type a file            (^T).
10255 '       33000 Error routine.
```

APPENDIX A
GRAPS SOFTWARE PROGRAM LISTINGS

The following are listings of the individual programs or modules that make up the GRAPS software. The modules are listed in the order in which they appear in the code line number listing given in the *main* module. The GRAPS program is simply a concatenation of these modules without all of the comment lines.

Each module contains a documentation header that identifies the module by name, describes the module's function and operating characteristics, and lists the name of the author of the code. When modifying a particular module, you should include in the *Edit History* section of the documentation header your name, the date of modification, and a description of any changes.

Modules numbered 10000 through 36000 deal with system maintenance; e.g., displaying the main menu, changing the system setup. The modules numbered 40000 through 56000 are the graph plot and display routines. These are the "old versions"[1] of the plot routines that I have modified to work within the GRAPS environment. Additional graph plot/display routines should be added after these, possibly being numbered from 47000 and 57000 for the plot and display routines, respectively.

[1]See appendix B for listings of these versions of the software.

REFERENCES

1. Lawrence Livermore Laboratory, Numerical Electromagnetics Code (NEC) — Method of Moments, by G.J. Burke and A.J. Pogio, Jan 1981.
2. Microsoft Corp., Leading Edge PC BASIC User's Manual — Version 2.0, 1983.
3. System Development Corp., Interactive Graphics Utility For Army NEC Automation (IGUANA), May 1985.
4. Hewlett-Packard Corp., Interfacing and Programming Manual — HP 7470A Graphics Plotter, 1982.
5. Microsoft Corp., Leading Edge PC MS-DOS — Version 2.11, 1984.

4. Text labels follow the data. All labels consist of two pairs: a label position and a text label. The label position is a *row, col* pair that describes the location of the first character in the text label that follows. After the label position is the text label. This should be a string of less than 80 characters, and should not have any leading blanks (spaces).

11.0 ADDITIONAL FEATURES

I have listed below additional features that would be desirable in GRAPS. These are enhancements, not included in the project objective, that would make the program more versatile.

1. Additional graph types; for example, a spline curve-fitting type. This would include facilities for viewing, plotting, and printing. The more graph types available on the system, the more general the system becomes.
2. A faster print routine. The present method of dumping the screen image to the printer is simply too slow. Either a method should be used that would be much faster, or the screen dump should be spooled so that the user can go about doing other things while the graph is being printed.
3. A better label editing facility. Additional capabilities should be introduced such as allowing the user to change existing labels. Also, it would be nice if just the label part of the graph could be saved to some file so that that set of labels could be "called up" and "pasted" over other graphs.

12.0 MODIFYING THE SYSTEM

Additions and modifications should be relatively easy to make once the organization of the system is understood. Each program module includes a documentation header that details certain aspects of the code and can be used as a guide in understanding and modifying that component of the system. In addition, comments narrate each action as it is performed. Understanding any aspect of the GRAPS system is as easy as reading the documentation and the code in the section of interest.

You must be careful, however, when changing any variable in any module since, when using BASIC, all is global. This can lead to disaster. I have tried to include in all module documentation a list of the global variables that are affected in that module.

Modularity of the components of the program offers the usual advantages in maintaining and upgrading the system. A module can easily be added by modifying only a few lines of code. For example, to add a module to plot log-log graphs, all you would have to do is write the subroutine to plot the graph and then change a few lines (about three) of existing code to accommodate the addition.

13.0 ADDITIONAL INFORMATION

Documentation abounds. I have kept track of my daily progress (in all areas) in my *Activity Log.*[10] I have kept the original programs that I modified to obtain the current versions of the plotting routines, as well as the sample data files, resultant plots, and screen dumps. In addition, I have plotted text/label character positions that can be used to position text on plots. Please note the documentation at the beginning of and throughout the program modules.

[10]Activity log for Robin Laird, generated at NOSC, San Diego, CA, Feb-May 1985. Available to qualified requesters.

1. First we must execute the program. This is described in the *Operation* section, so we refer to that section for further information. If all is well (that is, if the system search path is set correctly and the GRAPS software is resident), the main screen will appear after a few seconds.

2. To begin, we need to input the name of our data file. This is done by using function 1. Press function key 1, **F1.** The following prompt will appear:

   Filename<name.DAT>?

   Input the name of the data file. The *.DAT* means that the default extension of the data file is *.DAT.* So, if your data file has this extension, you do not have to type that part of the file name; it will be supplied for you. Type a<**RETURN**> to enter the name.

3. Now we need to enter the type of graph that the data describes. That is, is it linear data, log-linear data, etc.? Function 2 does this. Press function key 2, **F2.** The following prompt will appear:

   Plot type<NONE>?

   Input the data graph type. Simply type in one of the graph type names. As a reminder, the available types are *linear, bilinear, loglinearh, loglinearv, polar, Smith,* and *loglog.* Type <**RETURN**> to enter the name. (Note that *NONE* indicates that there is no file type default.)

4. Now we are ready to process the data; that is, view, plot, and print the graph. To view the graph, use function 3 and press **F3.** The graph will be displayed upon the screen. When you are finished looking at the graph, press function key 10, **F10.**

5. To plot the graph, use function key 4. Make sure that the pens are in the plotter, and that the paper is properly loaded. Press function key 4, **F4.** The message *plotting* will appear at the bottom of the screen, and will disappear when the operation is complete.

6. To print the graph, use function key 5. Make sure that the printer has paper, and that it is *ONLINE.* Press function key 5, **F5.** The graph will be displayed on the screen, and the image will be dumped to the printer. **Warning:** The print operation takes a long time (about 10 minutes).

10.0 DATA FILES

There are a few things that should be mentioned concerning the format of the data files. You should use the sample files in appendix III at the back of this manual for reference to the required formats. Looking at these files may be enough. However, here is also a list of the "critical" components of any data file.

1. Graph types requiring min/max values should have the following values at the beginning. X min/max are first, followed by Y min/max. The types that require min/max values are *linear, bilinear* (this requires two sets of Y min/max values), *Loglinear* (requires only an X min/max), *polar* (requires both a degree and radius min/max), and loglog.

2. Data pairs (X/Y coordinates, etc.) should be separated by commas. The X valuc (independent) should be listed first, the Y value (dependent) second.

3. Data sets (sets of data that describe a single plot/line) must be separated with the data pair *1.234, −1.234.* The last data set (or the only data set if there is just one) must end with the pair *−1.234, −1.234.* See the example files.

7.8 SAVE SETUP

The opposite of the read setup function, the save setup function allows a user to save a setup to a specified file. Enter the name of the file in which you wish to save the current configuration parameters. There is a default file name extension of **.SYS**. Note that a lone **<RETURN>** will abort this function.

7.9 CHANGE SETUP

This function allows the user to change any or all of the configurable options. See Section 6.0 for a list of the options and their possible values. When this function is invoked, an option window is displayed beneath the function window. The available options and their current values are shown. The cursor is located under the first option, and can be moved to the next item by using the down arrow. To change the value of an option, simply position the cursor under that option and press function key **F1**. If the item is a boolean value (e.g., ON/OFF, TRUE/FALSE), it will be changed to its opposite. Otherwise, you will be prompted to enter the new value. When you have finished making changes, press function key **F10**. This will erase the option window and return you to the main function menu.

7.10 EXIT PROGRAM

This function exits the GRAPS program. You will be prompted to confirm the exit operation. Typing anything but **NO** (or any derivative thereof; e.g., **no**) will exit the program and return you to the operating system.

8.0 SHELL COMMANDS

In addition to the functions described above, there are a few "hidden" capabilities of the GRAPS system. Certain control characters are used to invoke MS-DOS system functions. These commands are called "shell" commands, since they use the BASIC **SHELL** command in their execution. These functions were originally used in the development and debugging stages of the program, and were left in simply because they were very useful. To invoke these functions, you must hold down the control key,**<CTRL>**, while typing the indicated key.

8.1 DIRECTORY (CTRL-D)

This command gives a listing for the given directory. You are prompted for the path of the directory to list. Enter the path to list or a carriage return for the current directory (the one under which you entered GRAPS).

8.2 CHANGE LOGGED DIRECTORY (CTRL-L)

This allows the user to change the currently logged directory. You are prompted for the path of the directory to log. Enter the path of the directory to log (change to) or**<RETURN>** to see what the currently logged directory is.

8.3 TYPE A FILE (CTRL-T)

This is simply the *MS-DOS type* command in disguise. You are prompted for the name of the file to type. Enter the file name or**<RETURN>** to abort.

9.0 A TYPICAL SESSION

Outlined below is a "typical" session using the GRAPS software. The steps describe, from beginning to end, how the GRAPS package can be used to view, plot, and print a graph of a function. All items that are typed by the user are in **bold face**, and all prompts/responses by the program are in *italics*. I am assuming that the computer is turned on, and that all peripherals (the printer and plotter) are connected and ready.

7.6.1 MOVING LABELS

To change the position of a label, perform the following steps:

1. Position the cursor over the first character of the label to be repositioned. Use the directional arrows to move the cursor up, down, left, or right as required.
2. Type function key **F1**. This "picks" a label to reposition. You should hear a confirmation beep. If you don't, you probably have not hit the first character of the label (watch out for leading blanks in the data file).
3. Position the cursor over the character where you want the picked label to go. This is the destination of the label. You cannot position a label over another label, and there must be room for the label on the chosen line.
4. Type function key **F2**. This moves the label to the above position. If all is well (see above), you will hear a confirmation beep.

7.6.2 INSERTING LABELS

To insert a new label, follow these steps:

1. Position the cursor over the space where you want to insert the label. Note: You cannot insert a label over an existing label. Use the directional arrows to move the cursor.
2. Type function key **F3**. The cursor will disappear, and a confirmation beep will sound. If you don't hear the beep, you are trying to insert the label in an invalid place.
3. Type in the new label. End the entry with a<**RETURN**>. After the entry is made, the cursor will be positioned over the first character of the new label, and a confirmation beep will sound.

7.6.3 DELETING LABELS

To delete an existing label:

1. Position the cursor over the label you want to delete. Remember that deleting a label is a serious undertaking. The directional arrows are used to position the cursor.
2. Type function key **F4**. The label will be erased, and a confirmation beep will be sounded. At this point it is too late to recover the label.

7.6.4 SAVING YOUR WORK

Repeat the above steps until the labels are as you want them. When finished, you can **save** the new label positions by using function key **F9**. The current positions of the labels as they appear on the screen will replace the positions currently saved in the chosen file. A new file is created with the same name as the original data file, but with the extension *.EDT*. This file will contain the same data as the original, but will have the updated (edited) label positions. After the save operation, you will hear a confirmation beep indicating that the new values have been written. Function key **F10** is used to **exit** the edit function. Note that exiting **(F10)** does not cause any changes you have made to be saved.

7.7 READ SETUP

Read setup reads the configuration options from a given file. The function will prompt you for the name of the file that contains the new configuration information. The read setup function allows you to have several customized setups that can be used according to specific hardware/software arrangements. Simply enter the name of the file that contains the setup data. Note that a lone carriage return will abort the function.

```
                    (P$ = "LOGLOG")
12515 '
12520 'All done, so return.
12525       RETURN
```

DFSETUP MODULE

```
13000 REM
13005 REM ************************************************************************
13010 REM *
13015 REM *  Routine Name:     dfsetup
13020 REM *
13025 REM *  Description   :     This routine reads the default set up
13030 REM *                  configuration parameters. These are
13035 REM *                  simply the values for certain system
13040 REM *                  variables.
13045 REM *
13050 REM *  To Call       :     GOSUB 13000
13055 REM *
13060 REM *  Globals       :     All of the user-configurable, system
13065 REM *                  setup variables, e.g., SYS.COLOR%, are
13070 REM *                  affected by this routine.
13075 REM *
13080 REM *  Edit History  :     1) Robin Laird 4/23/85
13085 REM *
13090 REM ************************************************************************
13095 REM
13100 '
13105 'Open default setup file for input, and read system variables. If we
13110 'can't find the default file, then simply go with current (init) values.
13115 'Notice the setting of SYS.ERROR% and SYS.DO.ERRORS%.
13120     SYS.ERROR% = FALSE%
13125     SYS.DO.ERRORS% = FALSE%
13130     OPEN SYS.SETUP$ FOR INPUT AS #1
13135     IF SYS.ERROR% THEN GOTO 13215
13140     INPUT #1, SYS.COLOR%
13145     LINE INPUT #1, SYS.PLOTTER$
13150     INPUT #1, SYS.NUM.PENS%
13155     LINE INPUT #1, SYS.PRINTER$
13160     LINE INPUT #1, SYS.DATA.EXT$
13165     LINE INPUT #1, SYS.EDIT.EXT$
13170     INPUT #1, SYS.STAT.LINE%
13175     INPUT #1, SYS.SHELL%
13180     CLOSE #1
13185 '
13190 'Turn error processing back on, and return.
13195     SYS.DO.ERRORS% = TRUE%
13200     RETURN
13205 '
13210 'Turn error processing bak on, set sys variable to 'junk', and return.
13215     SYS.DO.ERRORS% = TRUE%
13220     SYS.SETUP$ = "NOTFOUND.SYS"
13225     RETURN
```

FKEYS MODULE

```
14000 REM
14005 REM ************************************************************************
14010 REM *
14015 REM *  Routine Name:     fkeys
14020 REM *
14025 REM *  Description   :     This subroutine sets up the function
14030 REM *                  keys. In particular, the function keys
14035 REM *                  are labeled "1" to "A" hexidecimal (note
14040 REM *                  that these are character strings). These
14045 REM *                  are "soft" labels that the function keys
14050 REM *                  will pass on to BASIC when pressed.
14055 REM *
14060 REM *  To Call       :     GOSUB 14000
14065 REM *
14070 REM *  Globals       :     None affected.
14075 REM *
14080 REM *  Edit History  :     1) Robin Laird 3/7/85
14085 REM *
14090 REM ************************************************************************
14095 REM
14100 '
14105 'Assign function key labels.
14110     KEY OFF
14115     FOR I = 1 TO NUM.FKEYS%
14120       KEY I, FKEY$(I)
14125     NEXT I
14130     RETURN
```

MENU MODULE

```
15000 REM
15005 REM ************************************************************************
15010 REM *
15015 REM *  Routine Name:     menu
15020 REM *
15025 REM *  Description   :     This subroutine displays the function
15030 REM *                  menu. A window is generated that shows
15035 REM *                  the available (function key) commands.
15040 REM *
15045 REM *  To Call       :     GOSUB 15000
15050 REM *
15055 REM *  Globals       :     The following variable(s) are affected
15060 REM *                  by this routine:
15065 REM *
15070 REM *                  BOX.pp - Parameters to the BOX, window
15075 REM *                       drawing, subroutine.
15080 REM *
15085 REM *  Edit History  :     1) Robin Laird 3/8/85
15090 REM *
15095 REM ************************************************************************
15100 REM
15105 '
15110 'Screen to page 0 in text mode (which means color).
15115     SCREEN 0,0,0,0
15120     WIDTH 80
15125     CLS
15130 '
15135 'Draw function "window"; in white.
15140     IF SYS.COLOR% THEN COLOR 7,0
15145     BOX.X1 = HSCRN.X.MIN : BOX.Y1 = HSCRN.Y.MIN
15150     BOX.X2 = HSCRN.X.MAX : BOX.Y2 = HSCRN.Y.MAX
15155     GOSUB 17000 'call box(x1, y1, x2, y2)
15160 '
15165 'Display screen title; in blue.
15170     IF SYS.COLOR% THEN COLOR 1,0
15175     LOCATE BOX.Y1-1,1
15180     PRINT FNSTRCNTR$("Function Menu", 80);
15185 '
15190 'Display menu in window; in green.
15195     IF SYS.COLOR% THEN COLOR 2,0
15200     FOR I = 1 TO NUM.MENU.LINES%
15205       LOCATE HSCRN.Y.MIN+I, HSCRN.X.MIN+2
15210       PRINT MENU.LINE$(I);
15215     NEXT I
15220 '
15225 'Print statistics info on 25'th line.
15230     IF SYS.COLOR% AND SYS.STAT.LINE% THEN COLOR 0,5 ELSE COLOR 7,0
15235     LOCATE SYS.STAT.LOCY, SYS.STAT.LOCX
15240     PRINT SPACE$(80);
15245     IF NOT SYS.STAT.LINE% THEN RETURN
15250     LOCATE SYS.STAT.LOCY, SYS.STAT.LOCX
15255     PRINT FNSTRCNTR$(SYS.FILENAME$,26); TAB(27);"|";
```

```
        FNSTRCNTR$(SYS.GRAPHTYPE$,26); TAB(54);"|";
        FNSTRCNTR$(SYS.SETUP$,26);
15260    RETURN
```

CLEAR MODULE

```
16000 REM
16005 REM ************************************************************************
16010 REM *
16015 REM *  Routine Name:     clear
16020 REM *
16025 REM *  Description   :     This subroutine clears a certain part
16030 REM *                  of the screen (normally the "home"
16035 REM *                  screen").
16040 REM *
16045 REM *  To Call       :     GOSUB 16000
16050 REM *
16055 REM *  Globals       :     The following variable(s) are used as
16060 REM *                  parameters to this routine:
16065 REM *
16070 REM *                  CLEAR.FROM, CLEAR.TO - The top and
16075 REM *                       bottom lines of the part of
16080 REM *                       screen to clear. They must be
16085 REM *                       within the screen boundaries.
16090 REM *
16095 REM *  Edit History  :     1) Robin Laird 3/11/85
16100 REM *
16105 REM ************************************************************************
16110 REM
16115 '
16120 'Make Y1 = min(CLEAR.FROM, CLEAR.TO) and Y2 = max(CLEAR.FROM, CLEAR.TO).
16125     IF (CLEAR.FROM < CLEAR.TO) THEN Y1 = CLEAR.FROM ELSE Y1 = CLEAR.TO
16130     IF (CLEAR.FROM < CLEAR.TO) THEN Y2 = CLEAR.TO ELSE Y2 = CLEAR.FROM
16135 '
16140 'Clear indicated area of screen.
16145     LOCATE Y1,1
16150     FOR I = Y1 TO Y2
16155       PRINT SPACE$(80);
16160     NEXT I
16165     RETURN
```

BOX MODULE

```
17000 REM
17005 REM ************************************************************************
17010 REM *
17015 REM *  Routine Name:     box
17020 REM *
17025 REM *  Description   :     This subroutine draws a box on the
17030 REM *                   screen using the extended ASCII code
17035 REM *                   characters. This is particularly
17040 REM *                   useful for systems without graphics
17045 REM *                   capabilities or when using text mode.
17050 REM *
17055 REM *  To Call       :     GOSUB 17000
17060 REM *
17065 REM *  Globals       :     The following variable(s) are used
17070 REM *                   as parameters to this subroutine:
17075 REM *
17080 REM *                   BOX.X1, BOX.Y1 - The coordinates of
17085 REM *                        the upper left corner of the
17090 REM *                        box (1<=X1<= 79, 1<=Y1<=24).
17095 REM *
17100 REM *                   BOX.X2, BOX.Y2 - The coordinates of
17105 REM *                        the lower right corner of the
17110 REM *                        box (2<=X2<=80, 2<=Y2<=25).
17115 REM *
17120 REM *  Edit History  :     1) Robin Laird 3/7/85
17125 REM *
17130 REM ************************************************************************
17135 REM
17140      IF (BOX.X1 >= BOX.X2) OR (BOX.Y1 >= BOX.Y2) THEN RETURN
17145 '
17150 'Draw box corners.
17155      LOCATE BOX.Y1, BOX.X1 : PRINT CHR$(218);
17160      LOCATE BOX.Y1, BOX.X2 : PRINT CHR$(191);
17165      LOCATE BOX.Y2, BOX.X1 : PRINT CHR$(192);
17170      LOCATE BOX.Y2, BOX.X2 : PRINT CHR$(217);
17175 '
17180 'Draw horizontal components.
17185      LOCATE BOX.Y1, BOX.X1+1 : PRINT STRING$(BOX.X2-BOX.X1-1,196);
17190      LOCATE BOX.Y2, BOX.X1+1 : PRINT STRING$(BOX.X2-BOX.X1-1,196);
17195 '
17200 'Draw vertical components.
17205      FOR I = 1 TO BOX.Y2-BOX.Y1-1
17210        LOCATE BOX.Y1+I, BOX.X1 : PRINT CHR$(179);
17215        LOCATE BOX.Y1+I, BOX.X2 : PRINT CHR$(179);
17220      NEXT I
17225      RETURN
```

QUE MODULE

```
18000 REM
18005 REM ************************************************************************
18010 REM *
18015 REM *  Routine Name:     que
18020 REM *
18025 REM *  Description   :     This subroutine is used to queue or
18030 REM *                   prompt the user to type a certain key
18035 REM *                   before continuing execution. It is
18040 REM *                   used, for example, after an error has
18045 REM *                   occurred, and the programmer wishes
18050 REM *                   the user to acknowledge the fact.
18055 REM *
18060 REM *  To Call       :     GOSUB 18000
18065 REM *
18070 REM *  Globals       :     The following variable(s) are used as
18075 REM *                   parameters to this subroutine:
18080 REM *
18085 REM *                   QUE.WAIT% - A boolean indicating whe-
18090 REM *                        ther or not to wait for user
18095 REM *                        response. Note that if TRUE%
18100 REM *                        then the queue string is erased
18105 REM *                        upon user response, otherwise
18110 REM *                        it is not.
18115 REM *
18120 REM *                   QUE.STR$ - The queue string that is
18125 REM *                        printed on the SYS.QUE line.
18130 REM *
18135 REM *  Edit History  :     1) Robin Laird 3/12/85
18140 REM *
18145 REM ************************************************************************
18150 REM
18155 '
18160 'Locate cursor at SYS.QUE line, print queue, and wait for user response.
18165     IF SYS.COLOR% THEN COLOR 3+INTENSE%, 0
18170     LOCATE SYS.QUE.LOCY, 1
18175     PRINT SPACE$(80);
18180     LOCATE SYS.QUE.LOCY, 1
18185     PRINT FNSTRCNTR$(QUE.STR$, 80);
18190     WHILE (INKEY$ = "" AND QUE.WAIT%) : WEND
18195 '
18200 'If we waited for a response then erase queue string else don't bother.
18205     LOCATE SYS.QUE.LOCY, 1
18210     IF QUE.WAIT% THEN PRINT SPACE$(80);
18215     LOCATE SYS.PROMPT.LOCY, 1
18220     RETURN
```

SHELL MODULE

```
19000 REM
19005 REM ************************************************************************
19010 REM *
19015 REM *  Routine Name:     shell
19020 REM *
19025 REM *  Description   :     This subroutine issues system-related
19030 REM *                  commands. Currently it uses the "shell"
19035 REM *                  statement to invoke the MS-DOS shell
19040 REM *                  with a given command. It can, however,
19045 REM *                  be modified to perform similar actions
19050 REM *                  using BASIC commands.
19055 REM *
19060 REM *  To Call       :     GOSUB 19000
19065 REM *
19070 REM *  Globals       :     The following variable(s) are used as
19075 REM *                  parameters to this routine:
19080 REM *
19085 REM *                  SHELL.STR$ - A string containing the
19090 REM *                       prompt to give the user. This
19095 REM *                       should be set to "" if a prompt
19100 REM *                       is not desired.
19105 REM *
19110 REM *                  SHELL.INPUT$ - A string that WILL con-
19115 REM *                       tain the input from the user in
19120 REM *                       response to the above prompt.
19125 REM *
19130 REM *                  SHELL.CMND$ - A string containing the
19135 REM *                       MS-DOS command to be performed.
19140 REM *
19145 REM *  Edit History  :     1) Robin Laird 3/13/85
19150 REM *
19155 REM ************************************************************************
19160 REM
19165 '
19170 'Make sure all OK to proceed.
19175     IF NOT SYS.SHELL% THEN RETURN
19180 '
19185 'Change color, clear screen and prompt user for desired input.
19190     IF SYS.COLOR% THEN COLOR 7,0
19195     IF SHELL.STR$ = "" THEN GOTO 19230
19200     CLS
19205     LOCATE 2,1
19210     PRINT SHELL.STR$;
19215     INPUT; SHELL.INPUT$
19220 '
19225 'Clear screen and issue DOS command.
19230     SHELL SHELL.CMND$+SHELL.INPUT$
19235     LOCATE 25,1
19240     PRINT ""
19245 '
19250 'Queue user to continue, and return on response.
19255     QUE.WAIT% = TRUE% : QUE.STR$ = "type any key to continue"
```

```
19260     GOSUB 18000 'call que(wait, str)
19265 '
19270 'Indicate that we messed up the screen.
19275     SYS.SCRN.CHANGED% = TRUE%
19280     RETURN
```

GETFN MODULE

```
20000 REM
20005 REM ************************************************************************
20010 REM *
20015 REM *  Routine Name:     getfn
20020 REM *
20025 REM *  Description   :     This subroutine queries the user for
20030 REM *                  the data filename. Note that the file
20035 REM *                  is opened as unit number one (#1).
20040 REM *
20045 REM *  To Call       :     GOSUB 20000
20050 REM *
20055 REM *  Globals       :     The following variable(s) are affected
20060 REM *                  by this subroutine:
20065 REM *
20070 REM *                  SYS.FILENAME$ - The name of the data
20075 REM *                       file.
20080 REM *
20085 REM *                  SYS.ERROR% - Boolean indicating whe-
20090 REM *                       ther or not a user error has
20095 REM *                       occurred.
20100 REM *
20105 REM *  Edit History  :     1) Robin Laird 3/5/85
20110 REM *
20115 REM ************************************************************************
20120 REM
20125 '
20130 'Prompt for data filename, and loop until we get it right! Note that
20135 'SYS.ERROR% will indicate if, in the OPEN statement, an error has
20140 'occurred.
20145     IF SYS.COLOR% THEN COLOR 3, 0
20150     DONE% = FALSE%
20155     WHILE (NOT DONE%)
20160       SYS.ERROR% = FALSE%
20165       LOCATE SYS.PROMPT.LOCY, SYS.PROMPT.LOCX
20170       PRINT "Filename <name.";SYS.DATA.EXT$;">";
20175       INPUT; S$
20180       IF S$ = "" THEN GOTO 20220
20185       IF INSTR(S$, ".") = 0 THEN S$ = S$ + "." + SYS.DATA.EXT$
20190       OPEN S$ FOR INPUT AS #1
20195       CLOSE #1
20200       IF NOT SYS.ERROR% THEN DONE% = TRUE%
20205     WEND
20210 '
20215 'Erase prompt, return if bad input, set sys variable.
20220     LOCATE SYS.PROMPT.LOCY, 1
20225     PRINT SPACE$(80);
20230     IF NOT DONE% THEN RETURN
20235     SYS.FILENAME$ = S$
20240 '
20245 'If status line on then print file name at bottom of screen.
20250     IF NOT SYS.STAT.LINE% THEN RETURN
20255     IF SYS.COLOR% THEN COLOR 0,5
```

```
20260     LOCATE SYS.STAT.LOCY, SYS.STAT.LOCX
20265     PRINT FNSTRCNTR$(SYS.FILENAME$,26); TAB(27);"|";
          FNSTRCNTR$(SYS.GRAPHTYPE$,26); TAB(54);"|";
          FNSTRCNTR$(SYS.SETUP$,26);
20270     RETURN
```

GETGT MODULE

```
21000 REM
21005 REM ************************************************************************
21010 REM *
21015 REM *  Routine Name:     getgt
21020 REM *
21025 REM *  Description   :     This subroutine queries the user for
21030 REM *                  the plot type of the graph. See the
21035 REM *                  function, "ptok", for informa-
21040 REM *                  tion on the available graph types.
21045 REM *
21050 REM *  To Call       :     GOSUB 21000
21055 REM *
21060 REM *  Globals       :     The following variable(s) are affected
21065 REM *                  by this subroutine:
21070 REM *
21075 REM *                  SYS.GRAPHTYPE$ - The plot type of the
21080 REM *                       graph. See above for details.
21085 REM *
21090 REM *                  SYS.GRAPHTYPE% - The index of the graph
21095 REM *                       type in the array of types.
21100 REM *
21105 REM *  Edit History  :     1) Robin Laird 3/5/85
21110 REM *
21115 REM ************************************************************************
21120 REM
21125 '
21130 'Prompt for graph plot type, and loop until we get it right! Error 100
21135 'indicates a value out of range.
21140     IF SYS.COLOR% THEN COLOR 3, 0
21145     DONE% = FALSE%
21150     WHILE (NOT DONE%)
21155       LOCATE SYS.PROMPT.LOCY, SYS.PROMPT.LOCX
21160       PRINT "Plot type <NONE>";
21165       INPUT; S$
21170       IF S$ = "" THEN GOTO 21235
21175 '
21180 'Convert input to upper case.
21185       T$ = ""
21190       FOR I = 1 TO LEN(S$)
21195         C = ASC(MID$(S$, I, 1))
21200         IF C >= 97 AND C <= 122 THEN C = C - 32
21205         T$ = T$ + CHR$(C)
21210       NEXT I
21215       IF FNPLOT.TYPE.OK(T$) THEN DONE% = TRUE% ELSE ERROR 100
21220     WEND
21225 '
21230 'Erase prompt, return if bad input, set sys variable.
21235     LOCATE SYS.PROMPT.LOCY, 1
21240     PRINT SPACE$(80);
21245     IF NOT DONE% THEN RETURN
21250     SYS.GRAPHTYPE$ = T$
21255 '
```

```
21260 'Determine index of SYS.GRAPHTYPE$ in GRAPH.TYPE$ array.
21265     SYS.GRAPHTYPE% = 1
21270     WHILE (SYS.GRAPHTYPE$ <> GRAPH.TYPE$(SYS.GRAPHTYPE%))
21275       SYS.GRAPHTYPE% = SYS.GRAPHTYPE% + 1
21280     WEND
21285 '
21290 'If status line on then print graph type at bottom of screen.
21295     IF NOT SYS.STAT.LINE% THEN RETURN
21300     IF SYS.COLOR% THEN COLOR  0,5
21305     LOCATE SYS.STAT.LOCY, SYS.STAT.LOCX
21310     PRINT FNSTRCNTR$(SYS.FILENAME$,26); TAB(27);"|";
          FNSTRCNTR$(SYS.GRAPHTYPE$,26); TAB(54);"|";
          FNSTRCNTR$(SYS.SETUP$,26);
21315     RETURN
```

```
28785     INPUT; NEW.ITEM$
28790     LOCATE SYS.PROMPT.LOCY, 1
28795     PRINT SPACE$(80);
28800     RETURN
```

```
28515     CLEAR.FROM = SYS.PROMPT.LOCY : CLEAR.TO = SYS.QUE.LOCY
28520     GOSUB 16000 'call clear(from, to)
28525     DONE% = TRUE%
28530     RETURN
28535 '
28540 '
28545 'ROUTINE TO GO TO NEXT ITEM. First,....
28550     IF ITEM = 8 THEN ITEM = 1 ELSE ITEM = ITEM + 1
28555     IF ITEM > 4 THEN MY.R = ITEM - 4  ELSE MY.R = ITEM
28560     IF ITEM > 4 THEN MY.C = 30        ELSE MY.C = 1
28565     LOCATE 14+MY.R, 14+MY.C
28570     RETURN
28575 '
28580 '
28585 'ROUTINE TO PRINT VAR NAMES AND CONTENTS. Start with upper left var.
28590     LOCATE 14+1,14+1 'item 1
28595     PRINT "Color          : ";
28600     IF SYS.COLOR% THEN PRINT "TRUE "; ELSE PRINT "FALSE";
28605 '
28610     LOCATE 14+2,14+1 'item 2
28615     PRINT "Plotter        : ";
28620     IF SYS.PLOTTER$ = "" THEN PRINT "NONE   "; ELSE PRINT SYS.PLOTTER$;
28625 '
28630     LOCATE 14+3,14+1 'item 3
28635     PRINT "Number of pens : ";
28640     PRINT FNLSD$(STR$(SYS.NUM.PENS%));
28645 '
28650     LOCATE 14+4,14+1 'item 4
28655     PRINT "Printer        : ";
28660     IF SYS.PRINTER$ = "" THEN PRINT "NONE   "; ELSE PRINT SYS.PRINTER$;
28665 '
28670     LOCATE 14+1,14+30 'item 5
28675     PRINT "Data file ext  : ";
28680     PRINT SYS.DATA.EXT$;
28685 '
28690     LOCATE 14+2,14+30 'item 6
28695     PRINT "Setup file ext : ";
28700     PRINT SYS.SETUP.EXT$;
28705 '
28710     LOCATE 14+3,14+30 'item 7
28715     PRINT "Status line    : ";
28720     IF SYS.STAT.LINE% THEN PRINT "ON "; ELSE PRINT "OFF";
28725 '
28730     LOCATE 14+4,14+30 'item 8
28735     PRINT "Shell commands : ";
28740     IF SYS.SHELL% THEN PRINT "TRUE "; ELSE PRINT "FALSE";
28745 '
28750     LOCATE 14+MY.R, 14+MY.C
28755     RETURN
28760 '
28765 '
28770 'ROUTINE TO INPUT (TEXT) VARIABLE ITEMS. Simply do as in GETFN and GETGT.
28775     LOCATE SYS.PROMPT.LOCY, SYS.PROMPT.LOCX
28780     PRINT "New value ";
```

```
28255     GOSUB 15000 'call menu()
28260     RETURN 28125
28265 '
28270 'Get plotter type.
28275     GOSUB 28775 'call getstr(new.item)
28280     IF NEW.ITEM$ = "NONE" OR NEW.ITEM$ = "none" THEN SYS.PLOTTER$ = ""
28285     GOSUB 28590
28290     RETURN
28295 '
28300 'Get number of pens in plotter.
28305     GOSUB 28775 'call getstr(new.item)
28310     N = VAL(NEW.ITEM$)
28315     IF N > 0 AND N < 7 THEN SYS.NUM.PENS% = N
28320     GOSUB 28590
28325     RETURN
28330 '
28335 'Get printer type.
28340     GOSUB 28775 'call getstr(new.item)
28345     IF NEW.ITEM$ = "NONE" OR NEW.ITEM$ = "none" THEN SYS.PRINTER$ = ""
28350     GOSUB 28590
28355     RETURN
28360 '
28365 'Get data file extension.
28370     GOSUB 28775 'call getstr(new.item)
28375     SYS.DATA.EXT$ = LEFT$(NEW.ITEM$, 3)
28380     GOSUB 28590
28385     RETURN
28390 '
28395 'Get setup file extension.
28400     GOSUB 28775 'call getstr(new.item)
28405     SYS.SETUP.EXT$ = LEFT$(NEW.ITEM$, 3)
28410     GOSUB 28590
28415     RETURN
28420 '
28425 'Toggle on/off stat line.
28430     SYS.STAT.LINE% = NOT SYS.STAT.LINE%
28435     IF SYS.COLOR% AND SYS.STAT.LINE% THEN COLOR 0,5 ELSE COLOR 7,0
28440     LOCATE SYS.STAT.LOCY, SYS.STAT.LOCX
28445     PRINT SPACE$(80);
28450     IF NOT SYS.STAT.LINE% THEN GOTO 28465
28455     LOCATE SYS.STAT.LOCY, SYS.STAT.LOCX
28460     PRINT FNSTRCNTR$(SYS.FILENAME$,26); TAB(27);"|";
          FNSTRCNTR$(SYS.GRAPHTYPE$,26); TAB(54);"|";
          FNSTRCNTR$(SYS.SETUP$,26);
28465     GOSUB 28590
28470     RETURN
28475 '
28480 'Toggle on/off shell commands avail.
28485     SYS.SHELL% = NOT SYS.SHELL%
28490     GOSUB 28590
28495     RETURN
28500 '
28505 '
28510 'ROUTINE TO EXIT CHSETUP. Simply change state of DONE% var.
```

CHSETUP MODULE

```
28000 REM
28005 REM ************************************************************************
28010 REM *
28015 REM *  Routine Name:     chsetup
28020 REM *
28025 REM *  Description   :     This routine allows the user to re-
28030 REM *                  define the operating parameters of
28035 REM *                  the program. Several of the system
28040 REM *                  variables are displayed on the screen
28045 REM *                  and the user can then "edit" these
28050 REM *                  variables, putting in whatever values
28055 REM *                  he/she would like.
28060 REM *
28065 REM *  To Call       :     GOSUB 28000
28070 REM *
28075 REM *  Globals       :     All of the user-configurable, system
28080 REM *                  setup variables, e.g., SYS.COLOR%, are
28085 REM *                  by this routine.
28090 REM *
28095 REM *  Edit History  :     1) Robin Laird 3/26/85
28100 REM *
28105 REM ************************************************************************
28110 REM
28115 '
28120 'Draw variable window with indicated color.
28125     IF SYS.COLOR% THEN COLOR 7,0
28130     BOX.X1 = 14 : BOX.Y1 = 14
28135     BOX.X2 = 67 : BOX.Y2 = 19
28140     GOSUB 17000 'call box(x1, y1, x2, y2)
28145 '
28150 'Print variable names and current values in window.
28155     IF SYS.COLOR% THEN COLOR 2,0
28160     GOSUB 28590 'call printvars()
28165 '
28170 'Locate at first item, and prepare for changes.
28175     ITEM = 1
28180     LOCATE 14+1,14+1
28185     DONE% = FALSE%
28190     WHILE (NOT DONE%)
28195       C$ = ""
28200       WHILE (C$ = "") : C$ = INKEY$ : WEND
28205     IF C$ = FKEY$(1)  THEN ON ITEM GOSUB 28245, 28275, 28305, 28340,
                                 28370, 28400, 28430, 28485
28210       IF C$ = FKEY$(10) THEN GOSUB 28515
28215       IF C$ = DNARROW$  THEN GOSUB 28550 'call relocate()
28220     WEND
28225     RETURN
28230 '
28235 '
28240 'ROUTINES TO CHANGE ITEMS. First one is for avail color.
28245     SYS.COLOR% = NOT SYS.COLOR%
28250     IF NOT SYS.COLOR% THEN COLOR 7,0
```

SVSETUP MODULE

```
27000 REM
27005 REM ************************************************************************
27010 REM *
27015 REM *  Routine Name:     svsetup
27020 REM *
27025 REM *  Description   :     This subroutine saves a system setup,
27030 REM *                  i.e., the variable information of the
27035 REM *                  system, to a file for later retrieval.
27040 REM *
27045 REM *  To Call       :     GOSUB 27000
27050 REM *
27055 REM *  Globals       :     The variable SYS.ERROR% is changed.
27060 REM *
27065 REM *  Edit History  :     1) Robin Laird 3/27/85
27070 REM *
27075 REM ************************************************************************
27080 REM
27085 '
27090 'Prompt for setup filename, and loop until we get it right! Note that
27095 'SYS.ERROR% will indicate if, in the OPEN statement, an error has
27100 'occurred.
27105     IF SYS.COLOR% THEN COLOR 3,0
27110     DONE% = FALSE%
27115     WHILE (NOT DONE%)
27120       SYS.ERROR% = FALSE%
27125       LOCATE SYS.PROMPT.LOCY, SYS.PROMPT.LOCX
27130       PRINT "Filename <name.";SYS.SETUP.EXT$;">";
27135       INPUT; S$
27140       IF S$ = "" THEN GOTO 27240
27145       IF INSTR(S$, ".") = 0 THEN S$ = S$ + "." + SYS.SETUP.EXT$
27150       OPEN S$ FOR OUTPUT AS #1
27155       CLOSE #1
27160       IF NOT SYS.ERROR% THEN DONE% = TRUE%
27165     WEND
27170 '
27175 'Write variable information to file.
27180     OPEN S$ FOR OUTPUT AS #1
27185     PRINT #1, SYS.COLOR%
27190     PRINT #1, SYS.PLOTTER$
27195     PRINT #1, SYS.NUM.PENS%
27200     PRINT #1, SYS.PRINTER$
27205     PRINT #1, SYS.DATA.EXT$
27210     PRINT #1, SYS.SETUP.EXT$
27215     PRINT #1, SYS.STAT.LINE%
27220     PRINT #1, SYS.SHELL%
27225     CLOSE #1
27230 '
27235 'Erase prompt.
27240     LOCATE SYS.PROMPT.LOCY, 1
27245     PRINT SPACE$(80);
27250     RETURN
```

```
26260       INPUT #1, SYS.STAT.LINE%
26265       INPUT #1, SYS.SHELL%
26270       CLOSE #1
26275 '
26280 'If new sys setup file then display main screen with updated parameters.
26285       IF DONE% THEN GOSUB 15000 'call menu()
26290       RETURN
```

RDSETUP MODULE

```
26000 REM
26005 REM ************************************************************************
26010 REM *
26015 REM *  Routine Name:     rdsetup
26020 REM *
26025 REM *  Description   :     This subroutine reads a system setup,
26030 REM *                   i.e., the variable information of the
26035 REM *                   system, from a file, and continues
26040 REM *                   execution with the new parameters.
26045 REM *
26050 REM *  To Call       :     GOSUB 26000
26055 REM *
26060 REM *  Globals       :     All of the user-configurable, system
26065 REM *                   setup variables, e.g., SYS.COLOR%, are
26070 REM *                   affected by this routine.
26075 REM *
26080 REM *  Edit History  :     1) Robin Laird 3/27/85
26085 REM *
26090 REM ************************************************************************
26095 REM
26100 '
26105 'Prompt for setup filename, and loop until we get it right! Note that
26110 'SYS.ERROR% will indicate if, in the OPEN statement, an error has
26115 'occurred.
26120     IF SYS.COLOR% THEN COLOR 3,0
26125     DONE% = FALSE%
26130     WHILE (NOT DONE%)
26135       SYS.ERROR% = FALSE%
26140       LOCATE SYS.PROMPT.LOCY, SYS.PROMPT.LOCX
26145       PRINT "Filename <name.";SYS.SETUP.EXT$;">";
26150       INPUT; S$
26155       IF S$ = "" THEN GOTO 26195
26160       IF INSTR(S$, ".") = 0 THEN S$ = S$ + "." + SYS.SETUP.EXT$
26165       OPEN S$ FOR INPUT AS #1
26170       CLOSE #1
26175       IF NOT SYS.ERROR% THEN DONE% = TRUE%
26180     WEND
26185 '
26190 'Erase prompt, return if bad input, set sys variable.
26195     LOCATE SYS.PROMPT.LOCY, 1
26200     PRINT SPACE$(80);
26205     IF NOT DONE% THEN RETURN
26210     SYS.SETUP$ = S$
26215 '
26220 'Read variable information from file.
26225     OPEN SYS.SETUP$ FOR INPUT AS #1
26230     INPUT #1, SYS.COLOR%
26235     LINE INPUT #1, SYS.PLOTTER$
26240     INPUT #1, SYS.NUM.PENS%
26245     LINE INPUT #1, SYS.PRINTER$
26250     LINE INPUT #1, SYS.DATA.EXT$
26255     LINE INPUT #1, SYS.SETUP.EXT$
```

```
25800     FOR I = MX TO NUM.LABELS%
25805       LABEL.POS(1,I) = LABEL.POS(1,I+1)
25810       LABEL.POS(2,I) = LABEL.POS(2,I+1)
25815       LABEL.STR$(I) = LABEL.STR$(I+1)
25820     NEXT I
25825     NUM.LABELS% = NUM.LABELS% - 1
25830 '
25835 'Re-position cursor, and return.
25840     LOCATE CR, CC
25845     GET ((POS(I)-1)*8,(CSRLIN-1)*8) - (POS(I)*8-1,CSRLIN*8-1), CSR%
25850     PUT ((POS(I)-1)*8, (CSRLIN-1)*8), CSR%, PRESET
25855     RETURN 25260
25860 '
25865 '
25870 'ROUTINE TO SAVE LABELS. First, open the data files for input/output.
25875     OPEN SYS.FILENAME$ FOR INPUT AS #1
25880     OPEN FNFILENAME$(SYS.FILENAME$, "EDT") FOR OUTPUT AS #3
25885 '
25890 'Read input file until we hit the values flagging end of data.
25895     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
25900     WHILE (X <> SYS.FLAG.VALUE OR Y <> SYS.FLAG.VALUE)
25905       INPUT #1, X, Y
25910       PRINT #3, X, Y
25915     WEND
25920 '
25925 'Print the new label information.
25930     FOR I = 1 TO NUM.LABELS%
25935       PRINT #3, LABEL.POS(1,I), LABEL.POS(2,I)
25940       PRINT #3, LABEL.STR$(I)
25945     NEXT I
25950 '
25955 'Close both files, sound confirmation bell, and return.
25960     CLOSE
25965     PRINT BELL$;
25970     RETURN 25260
25975 '
25980 '
25985 'ROUTINE TO EXIT EDIT LABELS ROUTINE. Simply set DONE% to TRUE%.
25990     DONE% = TRUE%
25995     RETURN  25260
```

```
25530     PUT ((POS(I)-1)*8, (CSRLIN-1)*8), CSR%, PRESET
25535     MY.R = SR : MY.C = SC
25540     RETURN 25260
25545 '
25550 '
25555 'ROUTINE TO INSERT LABEL. Make sure we are not positioned over a label.
25560     MX = 0 : LX = 80-OS% : I = 1 : CR = CSRLIN : CC = POS(I)
25565     WHILE (MX = 0 AND I <= NUM.LABELS%)
25570       L=LEN(LABEL.STR$(I)) : LR=LABEL.POS(1,I) : LC=LABEL.POS(2,I)+OS%
25575       IF (CR=LR AND (CC>=LC AND CC<=LC+L)) THEN MX = I
25580       IF (CR=LR AND CC<LC) THEN IF LC-CC < LX THEN LX = LC-CC
25585       I = I + 1
25590     WEND
25595     IF MX <> 0 THEN RETURN 25260 ELSE PRINT BELL$;
25600 '
25605 'Input new label; label is terminated by carraige return. Adjust input
25610 'string to max length (LX).
25615     LINE INPUT; E$
25620     IF E$ = "" THEN RETURN 25260
25625     E$ = LEFT$(E$, LX)
25630 '
25635 'Inc number of labels, and insert new label in label list.
25640     NUM.LABELS% = NUM.LABELS% + 1
25645     LABEL.POS(1,NUM.LABELS%) = CR
25650     LABEL.POS(2,NUM.LABELS%) = CC-OS%
25655     LABEL.STR$(NUM.LABELS%) = E$
25660 '
25665 'Restore line upon which input was done (fix labels possibly typed over).
25670     FOR I = 1 TO NUM.LABELS%
25675       IF LABEL.POS(1,I) <> CR THEN GOTO 25690
25680       LOCATE LABEL.POS(1,I), LABEL.POS(2,I)+OS%
25685       PRINT LABEL.STR$(I);
25690     NEXT I
25695 '
25700 'Re-position cursor, and return.
25705     LOCATE CR, CC
25710     GET ((POS(I)-1)*8,(CSRLIN-1)*8) - (POS(I)*8-1,CSRLIN*8-1), CSR%
25715     PUT ((POS(I)-1)*8, (CSRLIN-1)*8), CSR%, PRESET
25720     RETURN 25260
25725 '
25730 '
25735 'ROUTINE TO DELETE LABEL. First, make sure we are on a label.
25740     MX = 0 : I = 1 : CR = CSRLIN : CC = POS(I)
25745     WHILE (MX = 0 AND I <= NUM.LABELS%)
25750       L=LEN(LABEL.STR$(I)) : LR=LABEL.POS(1,I) : LC=LABEL.POS(2,I)+OS%
25755       IF (CR=LR AND (CC>=LC AND CC<=LC+L)) THEN MX = I
25760       I = I + 1
25765     WEND
25770     IF MX = 0 THEN RETURN 25260 ELSE PRINT BELL$;
25775     PX = 0
25780 '
25785 'Erase label, and remove it from label list.
25790     LOCATE LABEL.POS(1,MX), LABEL.POS(2,MX)+OS%
25795     PRINT SPACE$(LEN(LABEL.STR$(MX)));
```

```
25260     WEND
25265 '
25270 '
25275 'ROUTINE FOR MAINTAINING CURSOR CHAR. First get rid of old cursor.
25280     PUT ((POS(I)-1)*8, (CSRLIN-1)*8), CSR%, PSET
25285 '
25290 'Next, relocate cursor and display it at new pos.
25295     LOCATE MY.R, MY.C
25300     GET ((POS(I)-1)*8,(CSRLIN-1)*8) - (POS(I)*8-1,CSRLIN*8-1), CSR%
25305     PUT ((POS(I)-1)*8, (CSRLIN-1)*8), CSR%, PRESET
25310     RETURN
25315 '
25320 '
25325 'ROUTINE FOR PICKING A LABEL. First, see if on label (PX=index of label).
25330     PX = 0 : I = 1 : CR = CSRLIN : CC = POS(I)
25335     WHILE (PX = 0 AND I <= NUM.LABELS%)
25340       IF CR=LABEL.POS(1,I) AND CC=LABEL.POS(2,I)+OS% THEN PX = I
25345       I = I + 1
25350     WEND
25355     IF PX = 0 THEN RETURN 25260 ELSE PRINT BELL$;
25360 '
25365 'Set PR,PC to the row,column of the "picked" label.
25370     PR = LABEL.POS(1,PX) : PC = LABEL.POS(2,PX)+OS%
25375     RETURN 25260
25380 '
25385 '
25390 'ROUTINE TO MOVE PICKED LABEL. First, make sure we have something picked.
25395     IF PX = 0 THEN RETURN 25260
25400 '
25405 'Make sure we are not positioned 'around' another label.
25410     MX = 0 : I = 1 : CR = CSRLIN : CC = POS(I) : L1 = LEN(LABEL.STR$(PX))
25415     IF CC+L1 > 80-OS% THEN RETURN 25260
25420     WHILE (MX = 0 AND I <= NUM.LABELS%)
25425       L2=LEN(LABEL.STR$(I)) : LR=LABEL.POS(1,I) : LC=LABEL.POS(2,I)+OS%
25430       IF (CR=LR AND (CC>=LC AND CC<=LC+L2)) THEN MX = I
25435       IF (CR=LR AND (CC<LC AND CC+L1>=LC) AND (I<>PX)) THEN MX = I
25440       I = I + 1
25445     WEND
25450     IF MX <> 0 THEN RETURN 25260 ELSE PRINT BELL$;
25455 '
25460 'Save row,col. Delete label at old pos, and insert label at new pos.
25465     SR = CSRLIN : SC = POS(I)
25470     LOCATE PR, PC
25475     PRINT SPACE$(LEN(LABEL.STR$(PX)));
25480     LOCATE SR, SC
25485     PRINT LABEL.STR$(PX);
25490 '
25495 'Re-locate to new label pos, and reset PR,PC. Save new row,col values.
25500     LOCATE SR, SC
25505     PR = SR : PC = SC
25510     LABEL.POS(1,PX) = PR : LABEL.POS(2,PX) = PC-OS%
25515 '
25520 'Display cursor at new pos.
25525     GET ((POS(I)-1)*8,(CSRLIN-1)*8) - (POS(I)*8-1,CSRLIN*8-1), CSR%
```

EDIT MODULE

```
25000 REM
25005 REM ************************************************************************
25010 REM *
25015 REM *  Routine Name:     edit
25020 REM *
25025 REM *  Description   :     This subroutine allows the user to edit
25030 REM *                  the labels of a graph. The cursor keys
25035 REM *                  are used to "pick" labels and move them
25040 REM *                  to other positions on the screen.
25045 REM *
25050 REM *  To Call       :     GOSUB 25000
25055 REM *
25060 REM *  Globals       :     LABEL.xx - The label editing variables.
25065 REM *                      LABEL.POS, LABEL.STR$, and
25070 REM *                      NUM.LABELS% are affected.
25075 REM *
25080 REM *  Edit History  :     1) Robin Laird 3/22/85
25085 REM *
25090 REM ************************************************************************
25095 REM
25100 '
25105 'Check and make sure we have a graph to edit.
25110     IF SYS.FILENAME$ = "" OR SYS.GRAPHTYPE$ = "" THEN RETURN
25115 '
25120 'Display the graph, and set label editing offset variable.
25125     GT% = SYS.GRAPHTYPE%
25130     ON GT% GOSUB 50000,51000,52000,53000,54000,55000,56000
25135     IF (GT% = 3) OR (GT% = 4) OR (GT = 5) THEN OS% = 9 ELSE OS% = 0
25140 '
25145 'Turn off windowing, position and display cursor char.
25150     WINDOW
25155     LOCATE 1,1+OS%
25160     GET ((POS(I)-1)*8,(CSRLIN-1)*8) - (POS(I)*8-1,CSRLIN*8-1), CSR%
25165     PUT ((POS(I)-1)*8, (CSRLIN-1)*8), CSR%, PRESET
25170     MY.R = 1 : MY.C = 1+OS%
25175 '
25180 'Loop until we get a key command; exit when function key 10 is typed.
25185     DONE% = FALSE%
25190     WHILE (NOT DONE%)
25195       E$ = ""
25200       WHILE (E$ = "") : E$ = INKEY$ : WEND
25205       IF E$ = FKEY$(1)  THEN GOSUB 25330 'call picklabel()
25210       IF E$ = FKEY$(2)  THEN GOSUB 25395 'call movelabel()
25215       IF E$ = FKEY$(3)  THEN GOSUB 25560 'call inslabel()
25220       IF E$ = FKEY$(4)  THEN GOSUB 25740 'call dellabel()
25225       IF E$ = FKEY$(9)  THEN GOSUB 25875 'call savelabel()
25230       IF E$ = FKEY$(10) THEN GOSUB 25990 'call exit()
25235       IF E$ = UPARROW$  THEN IF CSRLIN > 1  THEN MY.R = MY.R-1
25240       IF E$ = DNARROW$  THEN IF CSRLIN < 25 THEN MY.R = MY.R+1
25245       IF E$ = LFARROW$  THEN IF POS(I) > 1+OS%  THEN MY.C = MY.C-1
25250       IF E$ = RTARROW$  THEN IF POS(I) < 80-OS% THEN MY.C = MY.C+1
25255       GOSUB 25280 'call cursor()
```

PRINT MODULE

```
24000 REM
24005 REM ************************************************************************
24010 REM *
24015 REM *  Routine Name:     print
24020 REM *
24025 REM *  Description   :     This subroutine prints the contents of
24030 REM *                   the LEPC screen. It first prompts the
24035 REM *                   user to ready the printer; next, the
24040 REM *                   graph is displayed on the screen and
24045 REM *                   the screen image is printed. When the
24050 REM *                   print operation is complete, the user
24055 REM *                   is returned to the main menu.
24060 REM *
24065 REM *  To Call       :     GOSUB 24000
24070 REM *
24075 REM *  Globals       :     The following variable(s) are affected
24080 REM *                   by this routine:
24085 REM *
24090 REM *                   QUE.pp - Parameters to the QUE, user
24095 REM *                        prompting, routine.
24100 REM *
24105 REM *  Edit History  :     1) Robin Laird 3/21/85
24110 REM *
24115 REM ************************************************************************
24120 REM
24125 '
24130 'Make sure there is something to print.
24135     IF SYS.FILENAME$="" OR SYS.GRAPHTYPE$="" OR SYS.PRINTER$="" THEN RETUN
24140 '
24145 'Query user to ready printer.
24150     QUE.WAIT% = TRUE% : QUE.STR$ = "type any key when ready to print"
24155     GOSUB 18000 'call que(wait, str)
24160 '
24165 'Fixed-format plotting routines. Jump to routine according to type.
24170     ON SYS.GRAPHTYPE% GOSUB 50000,51000,52000,53000,54000,55000,56000
24175 '
24180 'Get rid of cursor (we don't want it in the print out).
24185     LOCATE,,0
24190 '
24195 'Assign PRTSC address of print screen routine, and then call it.
24200     PRTSC = VARPTR(PRTSC%(0))
24205     CALL PRTSC
24210     RETURN
```

PLOT MODULE

```
23000 REM
23005 REM ************************************************************************
23010 REM *
23015 REM *  Routine Name:     plot
23020 REM *
23025 REM *  Description   :     This subroutine simply routes control
23030 REM *                  to one     of the plotting routines ac-
23035 REM *                  cording to the graph type set by the
23040 REM *                  "GETGT" subroutine.
23045 REM *
23050 REM *  To Call       :     GOSUB 23000
23055 REM *
23060 REM *  Globals       :     The following variable(s) are affected
23065 REM *                  by this routine:
23070 REM *
23075 REM *                  QUE.pp - Parameters to the QUE, user
23080 REM *                       prompting, routine.
23085 REM *  Edit History  :     1) Robin Laird 3/19/85
23090 REM *
23095 REM ************************************************************************
23100 REM
23105 '
23110 'Check and make sure all OK to proceed.
23115     IF SYS.FILENAME$="" OR SYS.GRAPHTYPE$="" OR SYS.PLOTTER$="" THEN RETURN
23120 '
23125 'Open com port to plotter (as unit #2).
23130     OPEN "COM1:9600, S, 7, 1, RS, CS65535, DS, CD" AS #2
23135 '
23140 'Query user to ready plotter, and then let user know what is going on.
23145     QUE.WAIT% = TRUE% : QUE.STR$ = "type any key when ready to plot"
23150     GOSUB 18000 ' call que(wait, str)
23155     QUE.WAIT% = FALSE% : QUE.STR$ = "plotting"
23160     GOSUB 18000 ' call que(wait, str)
23165 '
23170 'Fixed-format plotting routines. Jump to routine according to type (I).
23175     ON SYS.GRAPHTYPE% GOSUB 40000,41000,42000,43000,44000,45000,46000
23180 '
23185 'Clear prompts, queues, and close off com port.
23190     CLEAR.FROM = SYS.PROMPT.LOCY : CLEAR.TO = SYS.QUE.LOCY
23195     GOSUB 16000 'call clear(from, to)
23200     CLOSE #2
23205     RETURN
```

VIEW MODULE

```
22000 REM
22005 REM ************************************************************************
22010 REM *
22015 REM *  Routine Name:     view
22020 REM *
22025 REM *  Description   :     This subroutine simply routes control
22030 REM *                  to one of the screen display routines-
22035 REM *                  according to the graph type set by the
22040 REM *                  "GETGT" subroutine.
22045 REM *
22050 REM *  To Call       :     GOSUB 22000
22055 REM *
22060 REM *  Globals       :     The following variable(s) are affected
22065 REM *                  by this routine:
22070 REM *
22075 REM *                  QUE.pp - Parameters to the QUE, user
22080 REM *                       prompting, routine.
22085 REM *
22090 REM *  Edit History  :     1) Robin Laird 3/20/85
22095 REM *
22100 REM ************************************************************************
22105 REM
22110 '
22115 'Check and make sure all OK to proceed.
22120     IF SYS.FILENAME$ = "" OR SYS.GRAPHTYPE$ = "" THEN RETURN
22125 '
22130 'Get rid of cursor (we don't want it in the picture).
22135     LOCATE,,0
22140 '
22145 'Fixed-format plotting routines. Jump to routine according to type.
22150     ON SYS.GRAPHTYPE% GOSUB 50000,51000,52000,53000,54000,55000,56000
22155 '
22160 'Wait for user response before returning to main sreen.
22165     WHILE INKEY$ <> FKEY$(10) : WEND
22170     RETURN
```

EXIT MODULE

```
29000 REM
29005 REM ************************************************************************
29010 REM *
29015 REM *  Routine Name:     exit
29020 REM *
29025 REM *  Description   :     This routine is responsible for doing
29030 REM *                  any required "house keeping" before
29035 REM *                  a system (DOS) exit.
29040 REM *
29045 REM *  To Call       :     GOSUB 29000
29050 REM *
29055 REM *  Globals       :     The following variable(s) are affected
29060 REM *                  by this routine:
29065 REM *
29070 REM *                  SYS.EXIT% - A boolean that indicates
29075 REM *                       whether or not to exit the
29080 REM *                       program.
29085 REM *
29090 REM *  Edit History  :     1) Robin Laird 3/11/85
29095 REM *
29100 REM ************************************************************************
29105 REM
29110 '
29115 'Query user for exit confirmation.
29120     IF SYS.COLOR% THEN COLOR 3, 0
29125     LOCATE SYS.PROMPT.LOCY, SYS.PROMPT.LOCX
29130     PRINT "Are you sure <YES>";
29135     INPUT; S$
29140     IF (S$ <> "") AND (S$ <> "yes") AND (S$ <> "YES") THEN GOTO 29175
29145 '
29150 'Close all files and set exit flag.
29155     CLOSE
29160     SYS.EXIT% = TRUE%
29165 '
29170 'Erase prompt and return.
29175     LOCATE SYS.PROMPT.LOCY, 1
29180     PRINT SPACE$(80);
29185     RETURN
```

DIR MODULE

```
30000 REM
30005 REM ************************************************************************
30010 REM *
30015 REM *  Routine Name:     dir
30020 REM *
30025 REM *  Description   :     This subroutine generates and displays
30030 REM *                  a listing of a specified directory.
30035 REM *                  The user is prompted for the directory
30040 REM *                  to list.
30045 REM *
30050 REM *  To Call       :     GOSUB 30000
30055 REM *
30060 REM *  Globals       :     The following variable(s) are global
30065 REM *                  to this routine:
30070 REM *
30075 REM *                  SHELL.pp - Parameters to the SHELL,
30080 REM *                       MS-DOS invocation, subroutine.
30085 REM *
30090 REM *  Edit History  :     1) Robin Laird 3/12/85
30095 REM *
30100 REM ************************************************************************
30105 REM
30110 '
30115 'Set SHELL parameters appropriately, and call routine.
30120     SHELL.STR$ = "Directory to list <LOGGED>" : SHELL.CMND$="dir "
30125     GOSUB 19000 'call shell(str, cmnd)
30130     RETURN
```

CHDIR MODULE

```
31000 REM
31005 REM ************************************************************************
31010 REM *
31015 REM *  Routine Name:     chdir
31020 REM *
31025 REM *  Description   :     This subroutine allows the user to
31030 REM *                  change the currently logged directory.
31035 REM *                  The user is prompted for the directory
31040 REM *                  he/she desires to log.
31045 REM *
31050 REM *  To Call       :     GOSUB 31000
31055 REM *
31060 REM *  Globals       :     The following variable(s) are global
31065 REM *                  to this routine:
31070 REM *
31075 REM *                  SHELL.pp - Parameters to the SHELL,
31080 REM *                       MS-DOS invocation, subroutine.
31085 REM *
31090 REM *  Edit History  :     1) Robin Laird 3/14/85
31095 REM *
31100 REM ************************************************************************
31105 REM
31110 '
31115 'Set SHELL parameters appropriately, and call routine.
31120     SHELL.STR$ = "Change logged directory to <LOGGED>" : SHELL.CMND$="cd "
31125     GOSUB 19000 'call shell(str, cmnd)
31130     RETURN
```

TYPE MODULE

```
32000 REM
32005 REM ************************************************************************
32010 REM *
32015 REM *  Routine Name:     type
32020 REM *
32025 REM *  Description   :     This subroutine allows the user to type
32030 REM *                  the contents of a file on the screen.
32035 REM *                  The user is prompted for the file that
32040 REM *                  he/she desires to type.
32045 REM *
32050 REM *  To Call       :     GOSUB 32000
32055 REM *
32060 REM *  Globals       :     The following variable(s) are global
32065 REM *                  to this routine:
32070 REM *
32075 REM *                  SHELL.pp - Parameters to the SHELL,
32080 REM *                       MS-DOS invocation, subroutine.
32085 REM *
32090 REM *  Edit History  :     1) Robin Laird 3/14/85
32095 REM *
32100 REM ************************************************************************
32105 REM
32110 '
32115 'Set SHELL parameters appropriately, and call routine.
32120     SHELL.STR$ = "File to type <NONE>" : SHELL.CMND$="type "
32125     GOSUB 19000 'call shell(str, cmnd)
32130     RETURN
```

ERROR MODULE

```
33000 REM
33005 REM ************************************************************************
33010 REM *
33015 REM *  Routine Name:     error
33020 REM *
33025 REM *  Description   :     This subroutine processes system errors.
33030 REM *                  System errors include predefined BASIC
33035 REM *                  errors as well as user-defined errors.
33040 REM *                  User-defined errors are numbered from
33045 REM *                  100 on up, and are listed below.
33050 REM *
33055 REM *  To Call       :     ONE ERROR GOTO 33000
33060 REM *
33065 REM *  Globals       :     The following variable(s) are affected
33070 REM *                  by this routine:
33075 REM *
33080 REM *                  SYS.DO.ERRORS% - A boolean variable in-
33085 REM *                       dicating whether or not pro-
33090 REM *                       cessing of errors is to occur.
33095 REM *
33100 REM *                  SYS.ERROR% - System var indicating
33105 REM *                       whether or not an error has
33110 REM *                       occurred.
33115 REM *
33120 REM *                  ERRMESS.pp - Parameters to the ERRMESS,
33125 REM *                       error message resolution, sub-
33130 REM *                       routine.
33135 REM *
33140 REM *                  CLEAR.pp - Parameters to the CLEAR,
33145 REM *                       window erasing, subroutine.
33150 REM *
33155 REM *  Edit History  :     1) Robin Laird 3/11/85
33160 REM *
33165 REM ************************************************************************
33170 REM
33175      RESET
33180      SYS.ERROR% = TRUE%
33185 '
33190 'Check error recognition variable. If TRUE% then trap errors, else don't.
33195      IF NOT SYS.DO.ERRORS% THEN RESUME NEXT
33200 '
33205 'Process those errors that we have anticipated; get error message.
33210      SCREEN 0,0,0,0
33215      ERRMESS.NUM = ERR
33220      GOSUB 34000 'call errmess(num, str)
33225 '
33230 'Print error message (in blinking red), and user queue (in cyan).
33235      IF SYS.COLOR% THEN COLOR 4+BLINK%,0
33240      LOCATE SYS.ERR.LOCY, SYS.ERR.LOCX
33245      PRINT FNSTRCNTR$(ERRMESS.STR$, 80);
33250      QUE.WAIT% = TRUE% : QUE.STR$ = "type any key to continue"
33255      GOSUB 18000 'call que(str)
```

```
33260 '
33265 'Clear error message, and return to line after error.
33270     CLEAR.FROM = SYS.ERR.LOCY : CLEAR.TO = SYS.PROMPT.LOCY
33275     GOSUB 16000 'call clear(from, to)
33280     IF SYS.COLOR% THEN COLOR 3, 0
33285     RESUME NEXT
```

ERRMESS MODULE

```
34000 REM
34005 REM ************************************************************************
34010 REM *
34015 REM *  Routine Name:     errmess
34020 REM *
34025 REM *  Description   :     This subroutine returns an error mes-
34030 REM *                  sage according to the contents of an
34035 REM *                  error variable, ERRMESS.NUM. Note that
34040 REM *                  BASIC errors are numbered from 1 to 76,
34045 REM *                  user-defined errors from 100 on up.
34050 REM *
34055 REM *  To Call       :     GOSUB 34000
34060 REM *
34065 REM *  Globals       :     The following variable(s) are used as
34070 REM *                  parameters to this routine:
34075 REM *
34080 REM *                  ERRMESS.NUM - The number of the error.
34085 REM *
34090 REM *                  ERRMESS.STR$ - The error message that
34095 REM *                       corresponds to the above error
34100 REM *                       or "UNDEFINED ERROR" if the
34105 REM *                       error is not recognizable.
34110 REM *
34115 REM *  Edit History  :     1) Robin Laird 3/11/85
34120 REM *
34125 REM ************************************************************************
34130 REM
34135 '
34140 'Set message to default.
34145     ERRMESS.STR$ = "UNDEFINED ERROR"
34150     IF (ERRMESS.NUM = 53) THEN ERRMESS.STR$ = "FILE NOT FOUND"
34155     IF (ERRMESS.NUM = 54) THEN ERRMESS.STR$ = "BAD FILE MODE"
34160     IF (ERRMESS.NUM = 58) THEN ERRMESS.STR$ = "FILE ALREADY EXISTS"
34165     IF (ERRMESS.NUM = 61) THEN ERRMESS.STR$ = "DISK FULL"
34170     IF (ERRMESS.NUM = 64) THEN ERRMESS.STR$ = "BAD FILE NAME"
34175     IF (ERRMESS.NUM = 67) THEN ERRMESS.STR$ = "TOO MANY FILES"
34180     IF (ERRMESS.NUM = 70) THEN ERRMESS.STR$ = "DISK WRITE PROTECT"
34185     IF (ERRMESS.NUM = 71) THEN ERRMESS.STR$ = "DISK NOT READY"
34190     IF (ERRMESS.NUM = 72) THEN ERRMESS.STR$ = "DISK MEDIA ERROR"
34195     IF (ERRMESS.NUM = 75) THEN ERRMESS.STR$ = "PATH/FILE ACCESS ERROR"
34200     IF (ERRMESS.NUM = 76) THEN ERRMESS.STR$ = "PATH NOT FOUND"
34205     IF (ERRMESS.NUM = 100) THEN ERRMESS.STR$ = "INVALID PLOT TYPE"
34210     RETURN
```

LABEL MODULE

```
35000 REM
35005 REM ************************************************************************
35010 REM *
35015 REM *  Routine Name:     label
35020 REM *
35025 REM *  Description   :     This routine reads the label data from
35030 REM *                  the file opened as unit #1, and then
35035 REM *                  stores the information in the LABEL
35040 REM *                  (global) variable.
35045 REM *
35050 REM *  To Call       :     GOSUB 35000
35055 REM *
35060 REM *  Globals       :     The following variable(s) are affected
35065 REM *                  by this routine:
35070 REM *
35075 REM *                  LABEL.xx - The label editing variables.
35080 REM *                       LABEL.POS, LABEL.STR$, and
35085 REM *                       NUM.LABELS% are affected.
35090 REM *
35095 REM *  Edit History  :     1) Robin Laird 3/24/85
35100 REM *
35105 REM ************************************************************************
35110 REM
35115 '
35120 'Read row,col and text (string) until we hit the end.
35125     NUM.LABELS% = 0
35130     WHILE (NOT EOF(1))
35135       NUM.LABELS% = NUM.LABELS% + 1
35140       INPUT #1, LABEL.POS(1, NUM.LABELS%), LABEL.POS(2, NUM.LABELS%)
35145       LINE INPUT #1, LABEL.STR$(NUM.LABELS%)
35150     WEND
35155     RETURN
```

GPRINT MODULE

```
36000 REM
36005 REM ************************************************************************
36010 REM *
36015 REM *  Routine Name:     gprint
36020 REM *
36025 REM *  Description   :     This routine displays numeric data as
36030 REM *                  graphics characters in the hi-res mode.
36035 REM *                  Only the chars 0123456789-+. can be
36040 REM *                  displayed, and only two char "fonts"
36045 REM *                  are currently supported.
36050 REM *
36055 REM *  To Call       :     GOSUB 36000
36060 REM *
36065 REM *  Globals       :     The following are used as parameters
36070 REM *                  to the routine:
36075 REM *
36080 REM *                  GPRINT.X, GPRINT.Y - The X,Y location
36085 REM *                       of the lower left corner of
36090 REM *                       the string to display.
36095 REM *
36100 REM *                  GPRINT.SET% - An integer specifying the
36105 REM *                       character set (font) to use.
36110 REM *
36115 REM *                  GPRINT.STR$ - The (numeric) string to
36120 REM *                       display.
36125 REM *
36130 REM *  Edit History  :     1) Robin Laird 5/13/85
36135 REM *
36140 REM ************************************************************************
36145 REM
36150 '
36155 'Display each character in the string, starting at the indicated place.
36160 'XP and YP are "running" print locations, CV% is current char value.
36165     WINDOW
36170     XP = GPRINT.X : YP = GPRINT.Y
36175     FOR GI = 1 TO LEN(GPRINT.STR$)
36180       CV% = ASC(MID$(GPRINT.STR$, GI, 1))
36185       IF CV% = ASC(".") THEN CV% = 3 + (ASC("0") - 4)
36190       IF CV% = ASC("-") THEN CV% = 1 + (ASC("0") - 4)
36195       IF CV% = ASC("+") THEN CV% = 2 + (ASC("0") - 4)
36200       CV% = CV% - ASC("0") + 4
36205       IF CV% < 0 THEN CV% = 0
36210       ON GPRINT.SET% GOSUB 36235, 36325
36215     NEXT GI
36220     RETURN
36225 '
36230 'Character set one - normal size.
36235     ON CV% GOSUB 36250, 36255, 36260, 36265, 36270, 36275, 36280,
             36285, 36290, 36295, 36300, 36305, 36310
36240     XP = XP + 8
36245     RETURN
36250     PUT (XP,YP),AMINUS% : RETURN
```

```
36255     PUT (XP,YP),APLUS% : RETURN
36260     PUT (XP,YP),APOINT% : RETURN
36265     PUT (XP,YP),A0%    : RETURN
36270     PUT (XP,YP),A1%    : RETURN
36275     PUT (XP,YP),A2%    : RETURN
36280     PUT (XP,YP),A3%    : RETURN
36285     PUT (XP,YP),A4%    : RETURN
36290     PUT (XP,YP),A5%    : RETURN
36295     PUT (XP,YP),A6%    : RETURN
36300     PUT (XP,YP),A7%    : RETURN
36305     PUT (XP,YP),A8%    : RETURN
36310     PUT (XP,YP),A9%    : RETURN
36315 '
36320 'Character set two - approximately 1/2 scale.
36325     ON CV% GOSUB 36340, 36345, 36350, 36355, 36360, 36365, 36370,
             36375, 36380, 36385, 36390, 36395, 36400
36330     XP = XP + 5
36335     RETURN
36340     PUT (XP,YP),SMINUS% : RETURN
36345     PUT (XP,YP),SPLUS% : RETURN
36350     PUT (XP,YP),SPOINT% : RETURN
36355     PUT (XP,YP),S0%    : RETURN
36360     PUT (XP,YP),S1%    : RETURN
36365     PUT (XP,YP),S2%    : RETURN
36370     PUT (XP,YP),S3%    : RETURN
36375     PUT (XP,YP),S4%    : RETURN
36380     PUT (XP,YP),S5%    : RETURN
36385     PUT (XP,YP),S6%    : RETURN
36390     PUT (XP,YP),S7%    : RETURN
36395     PUT (XP,YP),S8%    : RETURN
36400     PUT (XP,YP),S9%    : RETURN
```

LINEAR MODULE

```
40000 REM
40005 REM ************************************************************************
40010 REM *
40015 REM *  Routine Name:     linear
40020 REM *
40025 REM *  Description   :     This routine plots the graph specified
40030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
40035 REM *                  The graph plots X and Y linearly along
40040 REM *                  the respective axes.
40045 REM *
40050 REM *  To Call       :     GOSUB 40000
40055 REM *
40060 REM *  Globals       :     The following variable(s) are affected
40065 REM *                  by this routine:
40070 REM *
40075 REM *                  LABEL.pp - The parameters to the LABEL,
40080 REM *                       data file text/label extraction,
40085 REM *                       routine.
40090 REM *
40095 REM *  Edit History  :     1) Robin Laird 3/14/85
40100 REM *
40105 REM ************************************************************************
40110 REM
40115 '
40120 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
40125 'color, and line type (PC - pen color, LT - line type).
40130     PRINT #2, FNINIT.PLOT$(1000, 960, 9000, 7250)
40135     PRINT #2, FNCHAR.DIR$(1,0)
40140     PRINT #2, FNCHAR.SIZE$(.208, .269)
40145     PRINT #2, PEN.COLOR.DEF$
40150     PRINT #2, LINE.TYPE.DEF$
40155     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
40160 '
40165 'Re-open data file for input as unit #1.
40170     OPEN SYS.FILENAME$ FOR INPUT AS #1
40175 '
40180 'Read X min/max and Y min/max, and calculate X and Y range.
40185     INPUT #1, X1, X2, Y1, Y2
40190     X.RANGE = X2 - X1
40195     Y.RANGE = Y2 - Y1
40200 '
40205 'Draw graph boundaries (graph border).
40210     PRINT #2, FNSTART.FROMD$(FNCVTX(0), FNCVTY(0))
40215     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(1))
40220     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(1))
40225     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(0))
40230     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(0))
40235 '
40240 'Draw axes tic marks (the scaling marks on the graph border). Note tha'
40245 'this is done for both the upper/lower (left/right) sides of the border.
40250 'Also, the "; XT;" instructs the plotter to draw a "tic" at the point.
40255     FOR Y = 0 TO 1
```

```
40260       FOR X = 1 TO SYS.X.PART-1
40265         PRINT #2, FNSTART.FROMD$(FNCVTX(1/SYS.X.PART*X), FNCVTY(Y))+"; XT;"
40270       NEXT X
40275     NEXT Y
40280 '
40285 'Do the same for the Y axis....
40290     FOR X = 0 TO 1
40295       FOR Y = 1 TO SYS.Y.PART-1
40300         PRINT #2, FNSTART.FROMD$(FNCVTX(X), FNCVTY(1/SYS.Y.PART*Y))+"; YT;"
40305       NEXT Y
40310     NEXT X
40315 '
40320 'Draw scale (tic) numbers.
40325     FOR I = 0 TO SYS.X.PART
40330       PRINT #2, FNSTART.FROMU$(FNCVTX(1/SYS.X.PART*I), SYS.YB.LABEL)
40335       PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.X.PART*I*X.RANGE+X1)))
40340     NEXT I
40345 '
40350 'Do the same for the Y axis....
40355     FOR I = 0 TO SYS.Y.PART
40360       PRINT #2, FNSTART.FROMU$(SYS.XL.LABEL, FNCVTY(1/SYS.Y.PART*I))
40365       PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.Y.PART*I*Y.RANGE+Y1)))
40370     NEXT I
40375 '
40380 'Init X/Y vars, and read data points until we reach the data flags.
40385     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
40390     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 40465
40395       NEWLINE = 1
40400       PRINT #2, FNPEN.COLOR$(PC)
40405       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
40410       PRINT #2, FNLINE.TYPE$(LT)
40415       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
40420       INPUT #1, X, Y
40425       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 40445
40430         ON NEWLINE GOSUB 40530, 40555
40435         INPUT #1, X, Y
40440       GOTO 40425
40445     GOTO 40390
40450 '
40455 'Process label entries. Each consists of a label coordinate (row, col)
40460 'plus a label.
40465     GOSUB 35000 'call label()
40470     PRINT #2, PEN.COLOR.DEF$
40475     FOR I = 1 TO NUM.LABELS%
40480       PRINT #2, FNPHLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
40485       PRINT #2, FNLABEL$(LABEL.STR$(I))
40490     NEXT I
40495 '
40500 'Close data file, put pen away, and return.
40505     CLOSE #1
40510     PRINT #2, FNPEN.COLOR$(0)
40515     RETURN
40520 '
40525 'Subroutine for beginning of line. Note the change to NEWLINE.
```

```
40530     PRINT #2, FNSTART.FROMD$(FNCVTX((X-X1)/X.RANGE),FNCVTY((Y-Y1)/Y.RANGE))
40535     NEWLINE = 2
40540     RETURN
40545 '
40550 'Subroutine to simply plot the point.
40555     PRINT #2, FNDRAW.TO$(FNCVTX((X-X1)/X.RANGE),FNCVTY((Y-Y1)/Y.RANGE))
40560     RETURN
```

BILIN MODULE

```
41000 REM
41005 REM ************************************************************************
41010 REM *
41015 REM *  Routine Name:     bilin
41020 REM *
41025 REM *  Description   :     This routine plots the graph specified
41030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
41035 REM *                  The graph plots X and and Y linearly
41040 REM *                  along the respective axes. In addition,
41045 REM *                  Y is scaled (differently) on both sides.
41050 REM *
41055 REM *  To Call       :     GOSUB 41000
41060 REM *
41065 REM *  Globals       :     The following variable(s) are affected
41070 REM *                  by this routine:
41075 REM *
41080 REM *                  LABEL.pp - The parameters to the LABEL,
41085 REM *                       data file text/label extraction,
41090 REM *                       routine.
41095 REM *
41100 REM *  Edit History  :     1) Robin Laird 3/19/85
41105 REM *
41110 REM ************************************************************************
41115 REM
41120 '
41125 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
41130 'color, and line type (PC - pen color, LT - line type).
41135     PRINT #2, FNINIT.PLOT$(1000, 960, 9000, 7250)
41140     PRINT #2, FNCHAR.DIR$(1,0)
41145     PRINT #2, FNCHAR.SIZE$(.208, .269)
41150     PRINT #2, PEN.COLOR.DEF$
41155     PRINT #2, LINE.TYPE.DEF$
41160     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
41165 '
41170 'Re-open data file for input as unit #1.
41175     OPEN SYS.FILENAME$ FOR INPUT AS #1
41180 '
41185 'Read X min/max and Y min/max, and calculate X and Y range.
41190     INPUT #1, X1, X2, Y1, Y2, Y3, Y4
41195     X.RANGE = X2 - X1
41200     Y1.RANGE = Y2 - Y1
41205     Y2.RANGE = Y4 - Y3
41210 '
41215 'Draw graph boundaries (graph border).
41220     PRINT #2, FNSTART.FROMD$(FNCVTX(0), FNCVTY(0))
41225     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(1))
41230     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(1))
41235     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(0))
41240     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(0))
41245 '
41250 'Draw axes tic marks (the scaling marks on the graph border). Note that
41255 'this is done for both the upper/lower (left/right` sides of the border.
```

LOGLOG MODULE

```
46000 REM
46005 REM ************************************************************************
46010 REM *
46015 REM *  Routine Name:     loglog
46020 REM *
46025 REM *  Description   :     This routine plots the graph specified
46030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
46035 REM *                  The graph plots X and Y logarithmically
46040 REM *                  along their respective axes.
46045 REM *
46050 REM *  To Call       :     GOSUB 46000
46055 REM *
46060 REM *  Globals       :     The following variable(s) are affected
46065 REM *                  by this routine:
46070 REM *
46075 REM *                  LABEL.pp - The parameters to the LABEL,
46080 REM *                       data file text/label extraction,
46085 REM *                       routine.
46090 REM *
46095 REM *  Edit History  :     1) Robin Laird 5/3/85
46100 REM *
46105 REM ************************************************************************
46110 REM
46115 '
46120 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
46125 'color, and line type (PC - pen color, LT - line type).
46130     PRINT #2, FNINIT.PLOT$(2650, 1825, 7650, 6825)
46135     PRINT #2, FNCHAR.DIR$(1, 0)
46140     PRINT #2, FNCHAR.SIZE$(.208, .269)
46145     PRINT #2, PEN.COLOR.DEF$
46150     PRINT #2, LINE.TYPE.DEF$
46155     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
46160 '
46165 'Re-open data file for input as unit #1.
46170     OPEN SYS.FILENAME$ FOR INPUT AS #1
46175 '
46180 'Read X min/max and Y min/max, and calculate X and Y range and num cycles.
46185     INPUT #1, X1, X2, Y1, Y2
46190     X1 = FNLOG10(X1) : X2 = FNLOG10(X2)
46195     Y1 = FNLOG10(Y1) : Y2 = FNLOG10(Y2)
46200     X.RANGE = X2 - X1
46205     Y.RANGE = Y2 - Y1
46210     Z1% = ABS(X2 - X1) : IF Z1% = 0 THEN Z1% = 1
46215     Z2% = ABS(Y2 - Y1) : IF Z2% = 0 THEN Z2% = 1
46220 '
46225 'Calculate steps for FOR-NEXT loops.
46230     SX% = Z1%18 : IF Z1% MOD 18 <> 0 THEN SX% = SX% + 1
46235     SY% = Z2%16 : IF Z2% MOD 16 <> 0 THEN SY% = SY% + 1
46240 '
46245 'Draw graph boundaries (graph border).
46250     PRINT #2, FNSTART.FROMD$(FNCVTX(0), FNCVTY(0))
46255     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(1))
```

```
45530     PRINT #2, FNSTART.FROMD$(FNCVTXS(K,I), FNCVTYS(K,I))
45535     NEWLINE = 2
45540     RETURN
45545 '
45550 'Subroutine to simply plot the point.
45555     PRINT #2, FNDRAW.TO$(FNCVTXS(K,I), FNCVTYS(K,I))
45560     RETURN
```

```
45260 '... and now for the other side....
45265     X = -1 : Y = 1 : R = 90
45270     FOR A = 1 TO 2
45275       PRINT #2, FNSTART.FROMD$(FNCVTXP(-1), FNCVTYP(0))
45280       PRINT #2, FNPARC$(FNCVTXP(X), FNCVTYP(Y), R)
45285       Y = Y + 1 : R = R - 36.5
45290     NEXT A
45295 '
45300 '... and finally the bar down the middle.
45305     PRINT #2, FNSTART.FROMD$(FNCVTXP(-1), FNCVTYP(0))
45310     PRINT #2, FNDRAW.TO$(FNCVTXP(1), FNCVTYP(0))
45315 '
45320 'Label the reactance circles.
45325     PRINT #2, FNSTART.FROMU$(FNCVTXP(-.05), FNCVTYP(.96))
45330     PRINT #2, FNLABEL$("-1")
45335     PRINT #2, FNSTART.FROMU$(FNCVTXP(.49),FNCVTYP(.83))
45340     PRINT #2, FNLABEL$("-.5")
45345     PRINT #2, FNSTART.FROMU$(FNCVTXP(.95), FNCVTYP(.05))
45350     PRINT #2, FNLABEL$("0")
45355     PRINT #2, FNSTART.FROMU$(FNCVTXP(.49), FNCVTYP(-.79))
45360     PRINT #2, FNLABEL$(".5")
45365     PRINT #2, FNSTART.FROMU$(FNCVTXP(-.05), FNCVTYP(-.95))
45370     PRINT #2, FNLABEL$("1")
45375 '
45380 'Init K/I vars, and read data points until we reach the data flags.
45385     K = NOT SYS.FLAG.VALUE : I = NOT SYS.FLAG.VALUE
45390     IF K = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 45465
45395       NEWLINE = 1
45400       PRINT #2, FNPEN.COLOR$(PC)
45405       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
45410       PRINT #2, FNLINE.TYPE$(LT)
45415       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
45420       INPUT #1, K, I
45425       IF I = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 45445
45430         ON NEWLINE GOSUB 45530, 45555
45435         INPUT #1, K, I
45440       GOTO 45425
45445     GOTO 45390
45450 '
45455 'Process label entries. Each consists of a label coordinate (row, col)
45460 'plus a label.
45465     GOSUB 35000 'call label()
45470     PRINT #2, PEN.COLOR.DEF$
45475     FOR I = 1 TO NUM.LABELS%
45480       PRINT #2, FNPVLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
45485       PRINT #2, FNLABEL$(LABEL.STR$(I))
45490     NEXT I
45495 '
45500 'Close data file, put pen away, and return.
45505     CLOSE #1
45510     PRINT #2, FNPEN.COLOR$(0)
45515     RETURN
45520 '
45525 'Subroutine for beginning of line. Note the change to NEWLINE.
```

SMITH MODULE

```
45000 REM
45005 REM ************************************************************************
45010 REM *
45015 REM *  Routine Name:     smith
45020 REM *
45025 REM *  Description   :     This routine plots the graph specified
45030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
45035 REM *                  The graph plots X and Y using the smith
45040 REM *                  coordinate system.
45045 REM *
45050 REM *  To Call       :     GOSUB 45000
45055 REM *
45060 REM *  Globals       :     The following variable(s) are affected
45065 REM *                  by this routine:
45070 REM *
45075 REM *                  LABEL.pp - The parameters to the LABEL,
45080 REM *                       data file text/label extraction,
45085 REM *                       routine.
45090 REM *
45095 REM *  Edit History  :     1) Robin Laird 4/10/85
45100 REM *
45105 REM ************************************************************************
45110 REM
45115 '
45120 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
45125 'color, and line type (PC - pen color, LT - line type).
45130     PRINT #2, FNINIT.PLOT$(2650, 1825, 7650, 6825)
45135     PRINT #2, FNCHAR.DIR$(0, -1)
45140     PRINT #2, FNCHAR.SIZE$(.208, .269)
45145     PRINT #2, PEN.COLOR.DEF$
45150     PRINT #2, LINE.TYPE.DEF$
45155     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
45160 '
45165 'Re-open data file for input as unit #1.
45170     OPEN SYS.FILENAME$ FOR INPUT AS #1
45175 '
45180 'Draw circular axes (at values 0, .3, 1, 3).
45185     X = -.75 : Y = 0 : R = .25
45190     FOR C = 1 TO 4
45195       PRINT #2, FNSTART.FROMU$(FNCVTXP(X), FNCVTYP(Y))
45200       PRINT #2, FNPCIRCLE$(R*HP.Y.MAX/2)
45205       X = X + .25 : R = R + .25
45210     NEXT C
45215 '
45220 'Draw semi-circle axes (at values -1, -.5, 0, .5, 1)....
45225     X = -1 : Y = -1 : R = -90
45230     FOR A = 1 TO 2
45235       PRINT #2, FNSTART.FROMD$(FNCVTXP(-1), FNCVTYP(0))
45240       PRINT #2, FNPARC$(FNCVTXP(X), FNCVTYP(Y), R)
45245       Y = Y - 1 : R = R + 36.5
45250     NEXT A
45255 '
```

```
44260       IF I > 0 THEN S$ = "+" ELSE IF I = 0 THEN S$ = "  " ELSE S$ = ""
44265       PRINT #2, FNLABEL$(S$ + FNLSD$(STR$(I)))
44270       P = P + 760
44275     NEXT I
44280     PRINT #2, FNSTART.FROMU$(SYS.XP.LABEL-200, SYS.YP.LABEL-P+760)
44285     PRINT #2, FNLABEL$("DBI")
44290 '
44295 'Init A/R vars, and read data points until we reach the data flags.
44300     A = NOT SYS.FLAG.VALUE : R = NOT SYS.FLAG.VALUE
44305     IF A = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 44380
44310       NEWLINE = 1
44315       PRINT #2, FNPEN.COLOR$(PC)
44320       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
44325       PRINT #2, FNLINE.TYPE$(LT)
44330       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
44335       INPUT #1, A, R
44340       IF R = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 44360
44345         ON NEWLINE GOSUB 44445, 44480
44350         INPUT #1, A, R
44355       GOTO 44340
44360     GOTO 44305
44365 '
44370 'Process label entries. Each consists of a label coordinate (row, col)
44375 'plus a label.
44380     GOSUB 35000 'call label()
44385     PRINT #2, PEN.COLOR.DEF$
44390     FOR I = 1 TO NUM.LABELS%
44395       PRINT #2, FNPVLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
44400       PRINT #2, FNLABEL$(LABEL.STR$(I))
44405     NEXT I
44410 '
44415 'Close data file, put pen away, and return.
44420     CLOSE #1
44425     PRINT #2, FNPEN.COLOR$(0)
44430     RETURN
44435 '
44440 'Subroutine for beginning of line. Note the change to NEWLINE.
44445     A = A*PI/180
44450     IF R < -40 THEN R = 0 ELSE R = (R+40)/50
44455     PRINT #2, FNSTART.FROMD$(FNCVTXP(R*COS(A)),FNCVTYP(-1*R*SIN(A)))
44460     NEWLINE = 2
44465     RETURN
44470 '
44475 'Subroutine to simply plot the point.
44480     A = A*PI/180
44485     IF R < -40 THEN R = 0 ELSE R = (R+40)/50
44490     PRINT #2, FNDRAW.TO$(FNCVTXP(R*COS(A)),FNCVTYP(-1*R*SIN(A)))
44495     RETURN
```

POLAR MODULE

```
44000 REM
44005 REM ************************************************************************
44010 REM *
44015 REM *  Routine Name:     polar
44020 REM *
44025 REM *  Description   :     This routine plots the graph specified
44030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
44035 REM *                  The graph is plotted in the polar co-
44040 REM *                  ordinate system.
44045 REM *
44050 REM *  To Call       :     GOSUB 44000
44055 REM *
44060 REM *  Globals       :     The following variable(s) are affected
44065 REM *                  by this routine:
44070 REM *
44075 REM *                  LABEL.pp - The parameters to the LABEL,
44080 REM *                       data file text/label extraction,
44085 REM *                       routine.
44090 REM *
44095 REM *  Edit History  :     1) Robin Laird 4/3/85
44100 REM *
44105 REM ************************************************************************
44110 REM
44115 '
44120 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
44125 'color, and line type (PC - pen color, LT - line type).
44130     PRINT #2, FNINIT.PLOT$(2650, 1825, 7650, 6825)
44135     PRINT #2, FNCHAR.DIR$(0, -1)
44140     PRINT #2, FNCHAR.SIZE$(.208, .269)
44145     PRINT #2, PEN.COLOR.DEF$
44150     PRINT #2, LINE.TYPE.DEF$
44155     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
44160 '
44165 'Re-open data file for input as unit #1.
44170     OPEN SYS.FILENAME$ FOR INPUT AS #1
44175 '
44180 'Draw polar grid. First, 5 concentric circles from radii 0.2 to 1.0.
44185     PRINT #2, FNSTART.FROMU$(FNCVTXP(0), FNCVTYP(0))
44190     FOR R = .2 TO 1! STEP .2
44195       PRINT #2, FNPCIRCLE$(R*HP.Y.MAX/2)
44200     NEXT R
44205 '
44210 'Now for the "cross-hairs" (the grid axes).
44215     PRINT #2, FNSTART.FROMD$(FNCVTXP(0), FNCVTYP(-1))
44220     PRINT #2, FNDRAW.TO$(FNCVTXP(0), FNCVTYP(1))
44225     PRINT #2, FNSTART.FROMD$(FNCVTXP(-1), FNCVTYP(0))
44230     PRINT #2, FNDRAW.TO$(FNCVTXP(1), FNCVTYP(0))
44235 '
44240 'Draw the grid labels (these will be the same for all graphs).
44245     P = 0
44250     FOR I = -40 TO 10 STEP 10
44255       PRINT #2, FNSTART.FROMU$(SYS.XP.LABEL, SYS.YP.LABEL-P)
```

```
43530       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
43535       INPUT #1, X, Y
43540       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 43560
43545         ON NEWLINE GOSUB 43645, 43680
43550         INPUT #1, X, Y
43555       GOTO 43540
43560     GOTO 43505
43565 '
43570 'Process label entries. Each consists of a label coordinate (row, col)
43575 'plus a label.
43580     GOSUB 35000 'call label()
43585     PRINT #2, PEN.COLOR.DEF$
43590     FOR I = 1 TO NUM.LABELS%
43595       PRINT #2, FNPHLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
43600       PRINT #2, FNLABEL$(LABEL.STR$(I))
43605     NEXT I
43610 '
43615 'Close data file, put pen away, and return.
43620     CLOSE #1
43625     PRINT #2, FNPEN.COLOR$(0)
43630     RETURN
43635 '
43640 'Subroutine for beginning of line. Note the change to NEWLINE.
43645     X = FNCVTX((FNLOG10(X) - X1)/X.RANGE)
43650     Y = FNCVTY((Y - Y1)/Y.RANGE)
43655     PRINT #2, FNSTART.FROMD$(X,Y)
43660     NEWLINE = 2
43665     RETURN
43670 '
43675 'Subroutine to simply plot the point.
43680     X = FNCVTX((FNLOG10(X) - X1)/X.RANGE)
43685     Y = FNCVTY((Y - Y1)/Y.RANGE)
43690     PRINT #2, FNDRAW.TO$(X,Y)
43695     RETURN
```

```
43260 'Draw axes tic marks, first for the X-horizontal axis. Note that this
43265 'is done for both the upper/lower sides of the border. Also, the "; XT;"
43270 'instructs the plotter to draw a "tic" at the point.
43275     FOR Y = 0 TO 1
43280       FOR C = 1 TO Z
43285         FOR X = 2 TO 10
43290           IF (X=10 AND C=Z) THEN P = -1 ELSE P = FNLOG10(X * 10^(C-1)) / Z
43295           PRINT #2, FNSTART.FROMD$(FNCVTX(P), FNCVTY(Y))+"; XT;"
43300         NEXT X
43305       NEXT C
43310     NEXT Y
43315 '
43320 '....do the same for the Y-vertical axis.
43325     FOR X = 0 TO 1
43330       FOR Y = 1 TO SYS.X.PART-1
43335         PRINT #2, FNSTART.FROMD$(FNCVTX(X), FNCVTY(1/SYS.X.PART*Y))+"; YT;"
43340       NEXT Y
43345     NEXT X
43350 '
43355 'Draw scale (tic) numbers, first for the X-horizontal, bottom axis.
43360     FOR I = 0 TO Z
43365       PRINT #2, FNCHAR.SIZE$(.208, .269)
43370       PRINT #2, FNSTART.FROMU$(FNCVTX(I/Z)-75, SYS.YB.LABEL)
43375       PRINT #2, FNLABEL$("10")
43380       PRINT #2, FNCHAR.SIZE$(.11, .16)
43385       PRINT #2, FNSTART.FROMU$(FNCVTX(I/Z)+140, SYS.YB.LABEL+125)
43390       PRINT #2, FNLABEL$(FNLSD$(STR$(FIX(X1+SGN(X1)*.5)+I)))
43395     NEXT I
43400 '
43405 '....and for the X-horizontal, top axis....
43410     FOR I = 1 TO Z
43415       FOR X = 1 TO 9
43420         P = FNLOG10(X * 10^(I-1)) / Z
43425         PRINT #2, FNSTART.FROMU$(FNCVTX(P)-10, SYS.YT.LABEL-140)
43430         PRINT #2, FNLABEL$(FNLSD$(STR$(X)))
43435       NEXT X
43440     NEXT I
43445     PRINT #2, FNSTART.FROMU$(FNCVTX(1)-2, SYS.YT.LABEL-140)
43450     PRINT #2, FNLABEL$("1")
43455 '
43460 '....and now for the Y-vertical axis (note change in size).
43465     PRINT #2, FNCHAR.SIZE$(.208, .269)
43470     FOR I = 0 TO SYS.X.PART STEP 2
43475       PRINT #2, FNSTART.FROMU$(SYS.XL.LABEL, FNCVTY(1/SYS.X.PART*I))
43480       PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.X.PART*I*Y.RANGE+Y1)))
43485     NEXT I
43490 '
43495 'Init X/Y vars, and read data points until we reach the data flags.
43500     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
43505     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 43580
43510       NEWLINE = 1
43515       PRINT #2, FNPEN.COLOR$(PC)
43520       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
43525       PRINT #2, FNLINE.TYPE$(LT)
```

LOGLINH MODULE

```
43000 REM
43005 REM ************************************************************************
43010 REM *
43015 REM *  Routine Name:     loglinh
43020 REM *
43025 REM *  Description   :     This routine plots the graph specified
43030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
43035 REM *                  The graph plots Y linearly along the
43040 REM *                  vertical axis, and X logarithmically
43045 REM *                  along the horizontal axis. The graph is
43050 REM *                  "horizontally" oriented.
43055 REM *
43060 REM *  To Call       :     GOSUB 43000
43065 REM *
43070 REM *  Globals       :     The following variable(s) are affected
43075 REM *                  by this routine:
43080 REM *
43085 REM *                  LABEL.pp - The parameters to the LABEL,
43090 REM *                       data file text/label extraction,
43095 REM *                       routine.
43100 REM *
43105 REM *  Edit History  :     1) Robin Laird 5/2/85
43110 REM *
43115 REM ************************************************************************
43120 REM
43125 '
43130 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
43135 'color, and line type (PC - pen color, LT - line type).
43140     PRINT #2, FNINIT.PLOT$(2650, 1825, 7650, 6825)
43145     PRINT #2, FNCHAR.DIR$(1, 0)
43150     PRINT #2, FNCHAR.SIZE$(.208, .269)
43155     PRINT #2, PEN.COLOR.DEF$
43160     PRINT #2, LINE.TYPE.DEF$
43165     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
43170 '
43175 'Re-open data file for input as unit #1.
43180     OPEN SYS.FILENAME$ FOR INPUT AS #1
43185 '
43190 'Read X min/max and Y min/max, and calculate X and Y range and num cycles.
43195     INPUT #1, X1, X2, Y1, Y2
43200     X1 = FNLOG10(X1) : X2 = FNLOG10(X2)
43205     X.RANGE = X2 - X1
43210     Y.RANGE = Y2 - Y1
43215     Z = ABS(FIX(X2 - X1)) : IF Z <= 0 THEN Z = 1
43220 '
43225 'Draw graph boundaries (graph border).
43230     PRINT #2, FNSTART.FROMD$(FNCVTX(0), FNCVTY(0))
43235     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(1))
43240     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(1))
43245     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(0))
43250     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(0))
43255 '
```

```
42530     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 42605
42535       NEWLINE = 1
42540       PRINT #2, FNPEN.COLOR$(PC)
42545       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
42550       PRINT #2, FNLINE.TYPE$(LT)
42555       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
42560       INPUT #1, X, Y
42565       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 42585
42570         ON NEWLINE GOSUB 42670, 42705
42575         INPUT #1, X, Y
42580       GOTO 42565
42585     GOTO 42530
42590 '
42595 'Process label entries. Each consists of a label coordinate (row, col)
42600 'plus a label.
42605     GOSUB 35000 'call label()
42610     PRINT #2, PEN.COLOR.DEF$
42615     FOR I = 1 TO NUM.LABELS%
42620       PRINT #2, FNPVLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
42625       PRINT #2, FNLABEL$(LABEL.STR$(I))
42630     NEXT I
42635 '
42640 'Close data file, put pen away, and return.
42645     CLOSE #1
42650     PRINT #2, FNPEN.COLOR$(0)
42655     RETURN
42660 '
42665 'Subroutine for beginning of line. Note the change to NEWLINE.
42670     X = FNCVTX((FNLOG10(X) - X1)/X.RANGE)
42675     Y = FNCVTY(1 - (Y - Y1)/Y.RANGE)
42680     PRINT #2, FNSTART.FROMD$(X,Y)
42685     NEWLINE = 2
42690     RETURN
42695 '
42700 'Subroutine to simply plot the point.
42705     X = FNCVTX((FNLOG10(X) - X1)/X.RANGE)
42710     Y = FNCVTY(1 - (Y - Y1)/Y.RANGE)
42715     PRINT #2, FNDRAW.TO$(X,Y)
42720     RETURN
```

```
42260     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(0))
42265     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(0))
42270 '
42275 'Draw axes tic marks, first for the Y-vertical axis. Note that this
42280 'is done for both the upper/lower (left/right) sides of the border.
42285 'Also, the "; XT;" instructs the plotter to draw a "tic" at the point.
42290     FOR Y = 0 TO 1
42295       FOR C = 1 TO Z% STEP SX%
42300         FOR X = 2 TO 10
42305           IF (X=10 AND C=Z%) THEN P=-1 ELSE P=FNLOG10(X * 10^(C-1))/Z%
42310           PRINT #2, FNSTART.FROMD$(FNCVTX(P), FNCVTY(Y))+"; XT;"
42315         NEXT X
42320       NEXT C
42325     NEXT Y
42330 '
42335 '....do the same for the X-horizontal axis.
42340     FOR X = 0 TO 1
42345       FOR Y = 1 TO SYS.X.PART-1
42350         PRINT #2, FNSTART.FROMD$(FNCVTX(X), FNCVTY(1/SYS.X.PART*Y))+"; YT;"
42355       NEXT Y
42360     NEXT X
42365 '
42370 'Draw scale (tic) numbers, first for the Y-vertical, left-hand axis (note
42375 'change in char size)....
42380     FOR I = 0 TO Z% STEP SX%
42385       PRINT #2, FNCHAR.SIZE$(.208, .269)
42390       PRINT #2, FNSTART.FROMU$(FNCVTX(I/Z%)-75, SYS.YT.LABEL+125)
42395       PRINT #2, FNLABEL$("10")
42400       PRINT #2, FNCHAR.SIZE$(.11, .16)
42405       PRINT #2, FNSTART.FROMU$(FNCVTX(I/Z%), SYS.YT.LABEL-125)
42410       PRINT #2, FNLABEL$(FNLSD$(STR$(FIX(X1+SGN(X1)*.5)+I)))
42415     NEXT I
42420 '
42425 '....and for the Y-vertical, right-hand axis (ONLY IF NUM CYCLES < 8)....
42430     IF Z% > 8 THEN GOTO 42490
42435     FOR I = 1 TO Z% STEP SX%
42440       FOR X = 1 TO 9
42445         P = FNLOG10(X * 10^(I-1))/Z%
42450         PRINT #2, FNSTART.FROMU$(FNCVTX(P)-10, SYS.YB.LABEL+75)
42455         PRINT #2, FNLABEL$(FNLSD$(STR$(X)))
42460       NEXT X
42465     NEXT I
42470     PRINT #2, FNSTART.FROMU$(FNCVTX(1)-2, SYS.YB.LABEL+75)
42475     PRINT #2, FNLABEL$("1")
42480 '
42485 '....and now for the X-horizontal axis (note change in size).
42490     PRINT #2, FNCHAR.SIZE$(.208, .269)
42495     FOR I = 0 TO SYS.X.PART STEP 2
42500       PRINT #2, FNSTART.FROMU$(SYS.XL.LABEL+300, FNCVTY(1/SYS.X.PART*I))
42505     PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.X.PART*(SYS.X.PART-I)*Y.RANGE+Y1)))
42510     NEXT I
42515 '
42520 'Init X/Y vars, and read data points until we reach the data flags.
42525     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
```

LOGLINV MODULE

```
42000 REM
42005 REM ************************************************************************
42010 REM *
42015 REM *  Routine Name:     loglinv
42020 REM *
42025 REM *  Description   :     This routine plots the graph specified
42030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
42035 REM *                  The graph plots X linearly along the
42040 REM *                  horizontal axis, and Y logarithmically
42045 REM *                  along the vertical axis. The graph is
42050 REM *                  "vertically" oriented.
42055 REM *
42060 REM *  To Call       :     GOSUB 42000
42065 REM *
42070 REM *  Globals       :     The following variable(s) are affected
42075 REM *                  by this routine:
42080 REM *
42085 REM *                  LABEL.pp - The parameters to the LABEL,
42090 REM *                       data file text/label extraction,
42095 REM *                       routine.
42100 REM *
42105 REM *  Edit History  :     1) Robin Laird 4/1/85
42110 REM *
42115 REM ************************************************************************
42120 REM
42125 '
42130 'Set plotter parameters, e.g., P1 and P2, char direction and size, pen
42135 'color, and line type (PC - pen color, LT - line type).
42140     PRINT #2, FNINIT.PLOT$(2650, 1825, 7650, 6825)
42145     PRINT #2, FNCHAR.DIR$(0, -1)
42150     PRINT #2, FNCHAR.SIZE$(.208, .269)
42155     PRINT #2, PEN.COLOR.DEF$
42160     PRINT #2, LINE.TYPE.DEF$
42165     PC = SYS.NUM.PENS% : LT = SYS.NUM.LINE.TYPES%-1
42170 '
42175 'Re-open data file for input as unit #1.
42180     OPEN SYS.FILENAME$ FOR INPUT AS #1
42185 '
42190 'Read X min/max and Y min/max, and calculate X and Y range and num cycles.
42195     INPUT #1, X1, X2, Y1, Y2
42200     X1 = FNLOG10(X1) : X2 = FNLOG10(X2)
42205     X.RANGE = X2 - X1
42210     Y.RANGE = Y2 - Y1
42215     Z% = ABS(FIX(X2 - X1)) : IF Z% <= 0 THEN Z% = 1
42220 '
42225 'Calculate steps for FOR-NEXT loops.
42230     SX% = Z%18 : IF Z% MOD 18 <> 0 THEN SX% = SX% + 1
42235 '
42240 'Draw graph boundaries (graph border).
42245     PRINT #2, FNSTART.FROMD$(FNCVTX(0), FNCVTY(0))
42250     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(1))
42255     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(1))
```

```
41530       PRINT #2, FNLABEL$(LABEL.STR$(I))
41535     NEXT I
41540 '
41545 'Close data file, put pen away, and return.
41550     CLOSE #1
41555     PRINT #2, FNPEN.COLOR$(0)
41560     RETURN
41565 '
41570 'Subroutine for beginning of line. Note the change to NEWLINE.
41575 'Also, note that the first graph will have the first range, the second...
41580     IF GNUM = 1 THEN Y.RANGE = Y1.RANGE ELSE Y.RANGE = Y2.RANGE
41585     PRINT #2, FNSTART.FROMD$(FNCVTX((X-X1)/X.RANGE),FNCVTY((Y-Y1)/Y.RANGE))
41590     GNUM = GNUM + 1
41595     NEWLINE = 2
41600     RETURN
41605 '
41610 'Subroutine to simply plot the point.
41615     PRINT #2, FNDRAW.TO$(FNCVTX((X-X1)/X.RANGE),FNCVTY((Y-Y1)/Y.RANGE))
41620     RETURN
```

```
41260 'Also, the "; XT;" instructs the plotter to draw a "tic" at the point.
41265     FOR Y = 0 TO 1
41270       FOR X = 1 TO SYS.X.PART-1
41275         PRINT #2, FNSTART.FROMD$(FNCVTX(1/SYS.X.PART*X), FNCVTY(Y))+"; XT;"
41280       NEXT X
41285     NEXT Y
41290 '
41295 'Do the same for the Y axis....
41300     FOR X = 0 TO 1
41305       FOR Y = 1 TO SYS.Y.PART-1
41310         PRINT #2, FNSTART.FROMD$(FNCVTX(X), FNCVTY(1/SYS.Y.PART*Y))+"; YT;"
41315       NEXT Y
41320     NEXT X
41325 '
41330 'Draw scale (tic) numbers.
41335     FOR I = 0 TO SYS.X.PART
41340       PRINT #2, FNSTART.FROMU$(FNCVTX(1/SYS.X.PART*I), SYS.YB.LABEL)
41345       PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.X.PART*I*X.RANGE+X1)))
41350     NEXT I
41355 '
41360 'Do the same for the Y axis....
41365     FOR I = 0 TO SYS.Y.PART
41370       PRINT #2, FNSTART.FROMU$(SYS.XL.LABEL, FNCVTY(1/SYS.Y.PART*I))
41375       PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.Y.PART*I*Y1.RANGE+Y1)))
41380     NEXT I
41385 '
41390 '... and for the right Y axis.
41395     FOR I = 0 TO SYS.Y.PART
41400       PRINT #2, FNSTART.FROMU$(SYS.XR.LABEL, FNCVTY(1/SYS.Y.PART*I))
41405       PRINT #2, FNLABEL$(FNLSD$(STR$(1/SYS.Y.PART*I*Y2.RANGE+Y1)))
41410     NEXT I
41415 '
41420 'Init X/Y and number of graph vars, and read data points until data flags.
41425     GNUM = 1
41430     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
41435     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 41510
41440       NEWLINE = 1
41445       PRINT #2, FNPEN.COLOR$(PC)
41450       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
41455       PRINT #2, FNLINE.TYPE$(LT)
41460       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
41465       INPUT #1, X, Y
41470       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 41490
41475         ON NEWLINE GOSUB 41580, 41615
41480         INPUT #1, X, Y
41485       GOTO 41470
41490     GOTO 41435
41495 '
41500 'Process label entries. Each consists of a label coordinate (row, col)
41505 'plus a label.
41510     GOSUB 35000 'call label()
41515     PRINT #2, PEN.COLOR.DEF$
41520     FOR I = 1 TO NUM.LABELS%
41525       PRINT #2, FNPHLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
```

```
46260     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(1))
46265     PRINT #2, FNDRAW.TO$(FNCVTX(1), FNCVTY(0))
46270     PRINT #2, FNDRAW.TO$(FNCVTX(0), FNCVTY(0))
46275 '
46280 'Draw axes tic marks, first for the X-horizontal axis. Note that this
46285 'is done for both the upper/lower sides of the border. Also, the "; XT;"
46290 'instructs the plotter to draw a "tic" at the point.
46295     FOR Y = 0 TO 1
46300       FOR C = 1 TO Z1% STEP SX%
46305         FOR X = 2 TO 10
46310           IF (X=10 AND C=Z1%) THEN P=-1 ELSE P=FNLOG10(X * 10^(C-1))/Z1%
46315           PRINT #2, FNSTART.FROMD$(FNCVTX(P), FNCVTY(Y))+"; XT;"
46320         NEXT X
46325       NEXT C
46330     NEXT Y
46335 '
46340 '....do the same for the Y-vertical axis.
46345     FOR X = 0 TO 1
46350       FOR C = 1 TO Z2% STEP SY%
46355         FOR Y = 2 TO 10
46360           IF (Y=10 AND C=Z2%) THEN P=-1 ELSE P=FNLOG10(Y * 10^(C-1))/Z2%
46365           PRINT #2, FNSTART.FROMD$(FNCVTX(X), FNCVTY(P))+"; YT;"
46370         NEXT Y
46375       NEXT C
46380     NEXT X
46385 '
46390 'Draw scale (tic) numbers, first for the X-horizontal, bottom axis.
46395     FOR I = 0 TO Z1% STEP SX%
46400       PRINT #2, FNCHAR.SIZE$(.208, .269)
46405       PRINT #2, FNSTART.FROMU$(FNCVTX(I/Z1%)-75, SYS.YB.LABEL)
46410       PRINT #2, FNLABEL$("10")
46415       PRINT #2, FNCHAR.SIZE$(.11, .16)
46420       PRINT #2, FNSTART.FROMU$(FNCVTX(I/Z1%)+140, SYS.YB.LABEL+125)
46425       PRINT #2, FNLABEL$(FNLSD$(STR$(FIX(X1+SGN(X1)*.5)+I)))
46430     NEXT I
46435 '
46440 '....and for the X-horizontal, top axis (ONLY IF NUM CYCLES < 8)....
46445     IF Z1% > 8 THEN GOTO 46505
46450     FOR I = 1 TO Z1% STEP SX%
46455       FOR X = 1 TO 9
46460         P = FNLOG10(X * 10^(I-1))/Z1%
46465         PRINT #2, FNSTART.FROMU$(FNCVTX(P)-10, SYS.YT.LABEL-140)
46470         PRINT #2, FNLABEL$(FNLSD$(STR$(X)))
46475       NEXT X
46480     NEXT I
46485     PRINT #2, FNSTART.FROMU$(FNCVTX(1)-2, SYS.YT.LABEL-140)
46490     PRINT #2, FNLABEL$("1")
46495 '
46500 '....and now for the Y-vertical axis, left side....
46505     FOR I = 0 TO Z2% STEP SY%
46510       PRINT #2, FNCHAR.SIZE$(.208, .269)
46515       PRINT #2, FNSTART.FROMU$(SYS.XL.LABEL, FNCVTY(I/Z2%)-75)
46520       PRINT #2, FNLABEL$("10")
46525       PRINT #2, FNCHAR.SIZE$(.11, .16)
```

```
46530       PRINT #2, FNSTART.FROMU$(SYS.XL.LABEL+225, FNCVTY(I/Z2%)+50)
46535       PRINT #2, FNLABEL$(FNLSD$(STR$(FIX(Y1+SGN(Y1)*.5)+I)))
46540     NEXT I
46545 '
46550 '....and so on with the right side (ONLY IF NUM CYCLES < 6)....
46555     IF Z2% > 6 THEN GOTO 46605
46560     FOR I = 1 TO Z2% STEP SY%
46565       FOR Y = 1 TO 9
46570         P = FNLOG10(Y * 10^(I-1))/Z2%
46575         PRINT #2, FNSTART.FROMU$(SYS.XR.LABEL, FNCVTY(P)-10)
46580         PRINT #2, FNLABEL$(FNLSD$(STR$(Y)))
46585       NEXT Y
46590     NEXT I
46595     PRINT #2, FNSTART.FROMU$(SYS.XR.LABEL, FNCVTY(1)-2)
46600     PRINT #2, FNLABEL$("1")
46605     PRINT #2, FNCHAR.SIZE$(.208, .269)
46610 '
46615 'Init X/Y vars, and read data points until we reach the data flags.
46620     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
46625     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 46700
46630       NEWLINE = 1
46635       PRINT #2, FNPEN.COLOR$(PC)
46640       IF PC <= 1 THEN PC = SYS.NUM.PENS% ELSE PC = PC - 1
46645       PRINT #2, FNLINE.TYPE$(LT)
46650       IF LT <= 1 THEN LT = SYS.NUM.LINE.TYPES%-1 ELSE LT = LT - 1
46655       INPUT #1, X, Y
46660       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 46680
46665         ON NEWLINE GOSUB 46765, 46800
46670         INPUT #1, X, Y
46675       GOTO 46660
46680     GOTO 46625
46685 '
46690 'Process label entries. Each consists of a label coordinate (row, col)
46695 'plus a label.
46700     GOSUB 35000 'call label()
46705     PRINT #2, PEN.COLOR.DEF$
46710     FOR I = 1 TO NUM.LABELS%
46715       PRINT #2, FNPHLOCATE$(LABEL.POS(1, I), LABEL.POS(2, I))
46720       PRINT #2, FNLABEL$(LABEL.STR$(I))
46725     NEXT I
46730 '
46735 'Close data file, put pen away, and return.
46740     CLOSE #1
46745     PRINT #2, FNPEN.COLOR$(0)
46750     RETURN
46755 '
46760 'Subroutine for beginning of line. Note the change to NEWLINE.
46765     X = FNCVTX((FNLOG10(X) - X1)/X.RANGE)
46770     Y = FNCVTY((FNLOG10(Y) - Y1)/Y.RANGE)
46775     PRINT #2, FNSTART.FROMD$(X,Y)
46780     NEWLINE = 2
46785     RETURN
46790 '
46795 'Subroutine to simply plot the point.
```

```
46800     X = FNCVTX((FNLOG10(X) - X1)/X.RANGE)
46805     Y = FNCVTY((FNLOG10(Y) - Y1)/Y.RANGE)
46810     PRINT #2, FNDRAW.TO$(X,Y)
46815     RETURN
```

LINEARV MODULE

```
50000 REM
50005 REM ************************************************************************
50010 REM *
50015 REM *  Routine Name:     linearv
50020 REM *
50025 REM *  Description   :     This routine displays the graph given
50030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
50035 REM *                  The graph displays X and Y linearly
50040 REM *                  along the respective axes.
50045 REM *
50050 REM *  To Call       :     GOSUB 50000
50055 REM *
50060 REM *  Globals       :     The following variable(s) are affected
50065 REM *                  by this routine:
50070 REM *
50075 REM *                  LABEL.pp - The parameters to the LABEL,
50080 REM *                       data file text/label extraction,
50085 REM *                       routine.
50090 REM *
50095 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
50100 REM *                       whether or not the screen has
50105 REM *                       changed (switched).
50110 REM *
50115 REM *  Edit History  :     1) Robin Laird 3/14/85
50120 REM *
50125 REM ************************************************************************
50130 REM
50135 '
50140 'Re-open data file for input as unit #1.
50145     OPEN SYS.FILENAME$ FOR INPUT AS #1
50150 '
50155 'Read x min/max and y min/max, and calculate x and y range.
50160     INPUT #1, X1, X2, Y1, Y2
50165     X.RANGE = X2 - X1
50170     Y.RANGE = Y2 - Y1
50175 '
50180 'Draw graph boundaries (graph border).
50185     SCREEN 2,0,0,0
50190     WINDOW (-.1, -.122) - (1.1, 1.065)
50195     LINE (0, 0) - (1, 1),,B
50200 '
50205 'Draw axes tic marks (the scaling marks on the graph border). Note that
50210 'this is done for both the upper/lower (left/right) sides of the border.
50215     FOR Y = 0 TO 1
50220       FOR X = 1 TO SYS.X.PART-1
50225         LINE (1/SYS.X.PART*X, Y-.012) - (1/SYS.X.PART*X, Y+.012)
50230       NEXT X
50235     NEXT Y
50240 '
50245 'Do the same for the Y axis....
50250     FOR X = 0 TO 1
50255       FOR Y = 1 TO SYS.Y.PART-1
```

```
50260         LINE (X-8.000001E-03,1/SYS.Y.PART*Y)-(X+8.000001E-03,1/SYS.Y.PART*Y)
50265       NEXT Y
50270     NEXT X
50275 '
50280 'Draw scale (tic) numbers.
50285     FOR I = 0 TO SYS.X.PART
50290       GPRINT.X = 1/SYS.X.PART*I*534+45 : GPRINT.Y = 184 : GPRINT.SET% = 1
50295       GPRINT.STR$ = STR$(1/SYS.X.PART*I*X.RANGE+X1)
50300       GOSUB 36000 'call gprint(x, y, s, str)
50305     NEXT I
50310 '
50315 'Do the same for the Y axis....
50320     FOR I = 0 TO SYS.Y.PART
50325       GPRINT.X = 16 : GPRINT.Y = 200-(1/SYS.Y.PART*I*168+24) : GPRINT.SET% = 1
50330       GPRINT.STR$ = STR$(1/SYS.Y.PART*I*Y.RANGE+Y1)
50335       GOSUB 36000 'call gprint(x, y, s, str)
50340     NEXT I
50345 '
50350 'Init X/Y vars, and read data points until we reach the data flags.
50355     WINDOW (-.1, -.122) - (1.1, 1.065)
50360     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
50365     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 50420
50370       NEWLINE = 1
50375       INPUT #1, X, Y
50380       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 50400
50385         ON NEWLINE GOSUB 50480, 50505
50390         INPUT #1, X, Y
50395       GOTO 50380
50400     GOTO 50365
50405 '
50410 'Process label entries. Each consists of a label coordinate (row, col)
50415 'plus a label.
50420     GOSUB 35000 'call label()
50425     FOR I = 1 TO NUM.LABELS%
50430       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)
50435       PRINT LABEL.STR$(I);
50440     NEXT I
50445 '
50450 'Close data file, set system screen indicator, and return.
50455     CLOSE #1
50460     SYS.SCRN.CHANGED% = TRUE%
50465     RETURN
50470 '
50475 'Subroutine for beginning of line. Note the change to NEWLINE.
50480     PSET ((X-X1)/X.RANGE, (Y-Y1)/Y.RANGE)
50485     NEWLINE = 2
50490     RETURN
50495 '
50500 'Subroutine to simply plot the point.
50505     LINE -((X-X1)/X.RANGE, (Y-Y1)/Y.RANGE)
50510     RETURN
```

BILINV MODULE

```
51000 REM
51005 REM ************************************************************************
51010 REM *
51015 REM *  Routine Name:     bilinv
51020 REM *
51025 REM *  Description   :     This routine displays the graph given
51030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
51035 REM *                  The graph displays X and and Y linearly
51040 REM *                  along the respective axes. In addition,
51045 REM *                  Y is scaled (differently) on both sides.
51050 REM *
51055 REM *  To Call       :     GOSUB 51000
51060 REM *
51065 REM *  Globals       :     The following variable(s) are affected
51070 REM *                  by this routine:
51075 REM *
51080 REM *                  LABEL.pp - The parameters to the LABEL,
51085 REM *                       data file text/label extraction,
51090 REM *                       routine.
51095 REM *
51100 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
51105 REM *                       whether or not the screen has
51110 REM *                       changed (switched).
51115 REM *
51120 REM *  Edit History  :     1) Robin Laird 3/22/85
51125 REM *
51130 REM ************************************************************************
51135 REM
51140 '
51145 'Re-open data file for input as unit #1.
51150     OPEN SYS.FILENAME$ FOR INPUT AS #1
51155 '
51160 'Read X min/max and Y min/max, and calculate X and Y range.
51165     INPUT #1, X1, X2, Y1, Y2, Y3, Y4
51170     X.RANGE = X2 - X1
51175     Y1.RANGE = Y2 - Y1
51180     Y2.RANGE = Y4 - Y3
51185 '
51190 'Draw graph boundaries (graph border).
51195     SCREEN 2,0,0,0
51200     WINDOW (-.1, -.122) - (1.1, 1.065)
51205     LINE (0, 0) - (1, 1),,B
51210 '
51215 'Draw axes tic marks (the scaling marks on the graph border). Note that
51220 'this is done for both the upper/lower (left/right) sides of the border.
51225     FOR Y = 0 TO 1
51230       FOR X = 1 TO SYS.X.PART-1
51235         LINE (1/SYS.X.PART*X, Y-.012) - (1/SYS.X.PART*X, Y+.012)
51240       NEXT X
51245     NEXT Y
51250 '
51255 'Do the same for the Y axis....
```

```
51260     FOR X = 0 TO 1
51265       FOR Y = 1 TO SYS.Y.PART-1
51270         LINE (X-8.000001E-03,1/SYS.Y.PART*Y)-(X+8.000001E-03,1/SYS.Y.PART*Y)
51275       NEXT Y
51280     NEXT X
51285 '
51290 'Draw scale (tic) numbers.
51295     FOR I = 0 TO SYS.X.PART
51300       GPRINT.X = 1/SYS.X.PART*I*534+45 : GPRINT.Y = 184 : GPRINT.SET% = 1
51305       GPRINT.STR$ = STR$(1/SYS.X.PART*I*X.RANGE+X1)
51310       GOSUB 36000 'call gprint(x, y, s, str)
51315     NEXT I
51320 '
51325 'Do the same for the left Y axis....
51330     FOR I = 0 TO SYS.Y.PART
51335       GPRINT.X = 16 : GPRINT.Y = 200-(1/SYS.Y.PART*I*168+24) : GPRINT.SET% = 1
51340       GPRINT.STR$ = STR$(1/SYS.Y.PART*I*Y1.RANGE+Y1)
51345       GOSUB 36000 'call gprint(x, y, s, str)
51350     NEXT I
51355 '
51360 '... and for the right Y axis.
51365     FOR I = 0 TO SYS.Y.PART
51370       GPRINT.X = 587 : GPRINT.Y = 200-(1/SYS.Y.PART*I*168+24) : GPRINT.SET% = 1
51375       GPRINT.STR$ = STR$(1/SYS.Y.PART*I*Y2.RANGE+Y3)
51380       GOSUB 36000 'call gprint(x, y, s, str)
51385     NEXT I
51390 '
51395 'Init X/Y and number of graph vars, and read data points until data flags.
51400     GNUM = 1
51405     WINDOW (-.1, -.122) - (1.1, 1.065)
51410     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
51415     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 51470
51420       NEWLINE = 1
51425       INPUT #1, X, Y
51430       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 51450
51435         ON NEWLINE GOSUB 51530, 51565
51440         INPUT #1, X, Y
51445       GOTO 51430
51450     GOTO 51415
51455 '
51460 'Process label entries. Each consists of a label coordinate (row, col)
51465 'plus a label.
51470     GOSUB 35000 'call label()
51475     FOR I = 1 TO NUM.LABELS%
51480       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)
51485       PRINT LABEL.STR$(I);
51490     NEXT I
51495 '
51500 'Close data file, set system screen indicator, and return.
51505     CLOSE #1
51510     SYS.SCRN.CHANGED% = TRUE%
51515     RETURN
51520 '
51525 'Subroutine for beginning of line. Note the change to NEWLINE.
```

```
51530     IF GNUM = 1 THEN Y.RANGE = Y1.RANGE ELSE Y.RANGE = Y2.RANGE
51535     PSET ((X-X1)/X.RANGE, (Y-Y1)/Y.RANGE)
51540     GNUM = GNUM + 1
51545     NEWLINE = 2
51550     RETURN
51555 '
51560 'Subroutine to simply plot the point.
51565     LINE -((X-X1)/X.RANGE, (Y-Y1)/Y.RANGE)
51570     RETURN
```

LOGLINVV MODULE

```
52000 REM
52005 REM ************************************************************************
52010 REM *
52015 REM *  Routine Name:     loglinvv
52020 REM *
52025 REM *  Description   :     This routine displays the graph given
52030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
52035 REM *                  The graph displays X linearly along the
52040 REM *                  horizontal axis, and Y logarithmically
52045 REM *                  along the vertical axis. The graph is
52050 REM *                  "vertically" oriented.
52055 REM *
52060 REM *  To Call       :     GOSUB 52000
52065 REM *
52070 REM *  Globals       :     The following variable(s) are affected
52075 REM *                  by this routine:
52080 REM *
52085 REM *                  LABEL.pp - The parameters to the LABEL,
52090 REM *                       data file text/label extraction,
52095 REM *                       routine.
52100 REM *
52105 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
52110 REM *                       whether or not the screen has
52115 REM *                       changed (switched).
52120 REM *
52125 REM *  Edit History  :     1) Robin Laird 4/10/85
52130 REM *
52135 REM ************************************************************************
52140 REM
52145 '
52150 'Re-open data file as unit #1.
52155     OPEN SYS.FILENAME$ FOR INPUT AS #1
52160 '
52165 'Read X min/max and Y min/max, and calculate X and Y range and num cycles.
52170     INPUT #1, X1, X2, Y1, Y2
52175     X1 = FNLOG10(X1) : X2 = FNLOG10(X2)
52180     X.RANGE = X2 - X1
52185     Y.RANGE = Y2 - Y1
52190     Z% = ABS(FIX(X2 - X1)) : IF Z% <= 0 THEN Z% = 1
52195 '
52200 'Calculate steps for FOR-NEXT loops (how many sets of tic marks, etc).
52205     SX% = Z%18     : IF Z% MOD 18 <> 0 THEN SX% = SX% + 1
52210     SY% = Y.RANGE500 : IF Y.RANGE MOD 500 <> 0 THEN SY% = SY% + 1
52215 '
52220 'Draw graph boundaries (graph border).
52225     SCREEN 2,0,0,0
52230     WINDOW (-.295, -.11) - (1.295, 1.05)
52235     LINE (0, 0) - (1, 1),,B
52240 '
52245 'Draw axes tic marks, first for the Y-vertical axis. Note that this
52250 'is done for both the upper/lower (left/right) sides of the border.
52255     FOR Y = 0 TO 1
```

```
52260       FOR C = 1 TO Z%
52265         FOR X = 2 TO 10
52270           IF (X=10 AND C=Z%) THEN P=-1 ELSE P=FNLOG10(X * 10^(C-1))/Z%
52275           LINE (Y-8.000001E-03, P) - (Y+8.000001E-03, P)
52280         NEXT X
52285       NEXT C
52290     NEXT Y
52295 '
52300 '....do the same for the X-horizontal axis.
52305     FOR X = 0 TO 1
52310       FOR Y = 1 TO SYS.X.PART-1
52315         LINE (1/SYS.X.PART*Y, X-.012) - (1/SYS.X.PART*Y, X+.012)
52320       NEXT Y
52325     NEXT X
52330 '
52335 'Draw scale (tic numbers), first for the Y-vertical, left-hand axis.
52340     FOR I = 0 TO Z% STEP SX%
52345       GPRINT.X = 80 : GPRINT.Y = 188-(I/Z%*168+8) : GPRINT.SET% = 1
52350       GPRINT.STR$ = "10"
52355       GOSUB 36000 'call gprint(x, y, s, str)
52360       GPRINT.X = 98 : GPRINT.Y = GPRINT.Y-6 : GPRINT.SET% = 2
52365       GPRINT.STR$ = FNLSD$(STR$(FIX(X1+SGN(X1)*.5)+I))
52370       GOSUB 36000 'call gprint(x, y, s, str)
52375     NEXT I
52380 '
52385 '....and now for the X-horizontal axis.
52390     FOR I = 0 TO SYS.X.PART STEP SY%
52395       GPRINT.X = 1/SYS.X.PART*I*400+118 : GPRINT.Y = 184 : GPRINT.SET% = 1
52400       GPRINT.STR$ = FNLSD$(STR$(1/SYS.X.PART*I*Y.RANGE+Y1))
52105       GOSUB 36000 'call gprint(x, y, s, str)
52410     NEXT I
52115 '
52420 'Init X/Y vars, and read data points until we reach the data flags.
52425     WINDOW (-.295, -.11) - (1.295, 1.05)
52430     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
52435     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 52490
52440       NEWLINE = 1
52145       INPUT #1, X, Y
52450       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 52470
52155         ON NEWLINE GOSUB 52550, 52585
52160         INPUT #1, X, Y
52165       GOTO 52450
52470     GOTO 52435
52475 '
52480 'Process label entries. Each consists of a label coordinate (row, col)
52485 'plus a label.
52490     GOSUB 35000 'call label()
52495     FOR I = 1 TO NUM.LABELS%
52500       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)+10
52505       PRINT LABEL.STR$(I);
52510     NEXT I
52515 '
52520 'Close data file, put pen away, and return.
52525     CLOSE #1
```

```
52530     SYS.SCRN.CHANGED% = TRUE%
52535     RETURN
52540 '
52545 'Subroutine for beginning of line. Note the change to NEWLINE.
52550     X = (FNLOG10(X) - X1)/X.RANGE
52555     Y = (Y - Y1)/Y.RANGE
52560     PSET (Y,X)
52565     NEWLINE = 2
52570     RETURN
52575 '
52580 'Subroutine to simply plot the point.
52585     X = (FNLOG10(X) - X1)/X.RANGE
52590     Y = (Y - Y1)/Y.RANGE
52595     LINE -(Y,X)
52600     RETURN
```

LOGLINHV MODULE

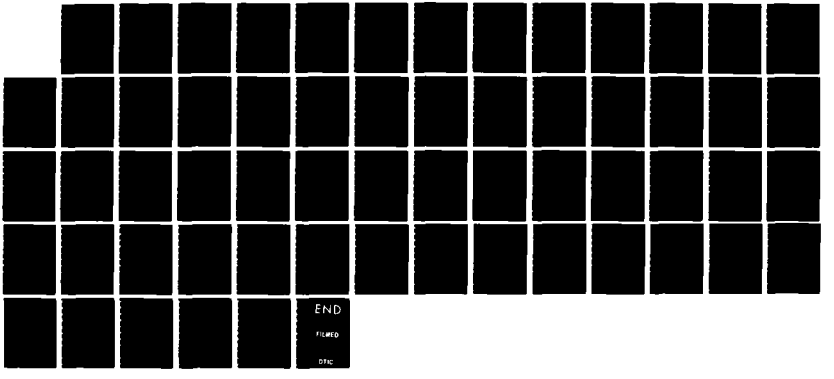
```
53000 REM
53005 REM ************************************************************************
53010 REM *
53015 REM *  Routine Name:     loglinhv
53020 REM *
53025 REM *  Description   :     This routine displays the graph given
53030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
53035 REM *                  The graph displays Y linearly along the
53040 REM *                  vertical axis, and X logarithmically
53045 REM *                  along the horizontal axis. The graph is
53050 REM *                  "horizontally" oriented. Note that the
53055 REM *                  small scale numbers that appear on the
53060 REM *                  left axis of the plot are not included
53065 REM *                  in the displayed version.
53070 REM *
53075 REM *  To Call       :     GOSUB 53000
53080 REM *
53085 REM *  Globals       :     The following variable(s) are affected
53090 REM *                  by this routine:
53095 REM *
53100 REM *                  LABEL.pp - The parameters to the LABEL,
53105 REM *                       data file text/label extraction,
53110 REM *                       routine.
53115 REM *
53120 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
53125 REM *                       whether or not the screen has
53130 REM *                       changed (switched).
53135 REM *
53140 REM *  Edit History  :     1) Robin Laird 4/10/85
53145 REM *
53150 REM ************************************************************************
53155 REM
53160 '
53165 'Re-open data file as unit #1.
53170     OPEN SYS.FILENAME$ FOR INPUT AS #1
53175 '
53180 'Read X min/max and Y   iin/max, and calculate X and Y range and num cycles.
53185     INPUT #1, X1, X2, Y1, Y2
53190     X1 = FNLOG10(X1) : X2 = FNLOG10(X2)
53195     X.RANGE = X2 - X1
53200     Y.RANGE = Y2 - Y1
53205     Z% = ABS(FIX(X2 - X1)) : IF Z% <= 0 THEN Z% = 1
53210 '
53215 'Calculate steps for FOR-NEXT loops (how many sets of tic marks, etc).
53220     SX% = Z%18     : IF Z% MOD 18 <> 0 THEN SX% = SX% + 1
53225     SY% = Y.RANGE500 : IF Y.RANGE MOD 500 <> 0 THEN SY% = SY% + 1
53230 '
53235 'Draw graph boundaries (graph border).
53240     SCREEN 2,0,0,0
53245     WINDOW (-.12, -.122) - (1.1, 1.065)
53250     LINE (0, 0) - (1, 1),,B
53255 '
```

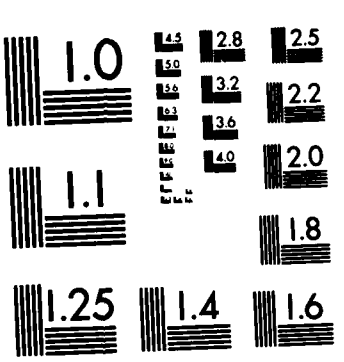
```
53260 'Draw axes tic marks, first for the X-horizontal axis. Note that this
53265 'is done for both the upper/lower sides of the border.
53270     FOR X = 0 TO 1
53275       FOR C = 1 TO Z% STEP SX%
53280         FOR Y = 2 TO 10
53285           IF (Y=10 AND C=Z%) THEN P=-1 ELSE P=FNLOG10(Y * 10^(C-1))/Z%
53290           LINE (P, X-8.000001E-03) - (P, X+8.000001E-03)
53295         NEXT Y
53300       NEXT C
53305     NEXT X
53310 '
53315 '....do the same for the Y-vertical axis.
53320     FOR X = 0 TO 1
53325       FOR Y = 1 TO SYS.X.PART-1
53330         LINE (X-.006, 1/SYS.X.PART*Y) - (X+.006 ,1/SYS.X.PART*Y)
53335       NEXT Y
53340     NEXT X
53345 '
53350 'Draw scale (tic) numbers, first for the X-horizontal, bottom axis.
53355     FOR I = 0 TO Z% STEP SX%
53360       GPRINT.X = I/Z%*520+60 : GPRINT.Y = 184 : GPRINT.SET% = 1
53365       GPRINT.STR$ = "10"
53370       GOSUB 36000 'call gprint(x, y, s, str)
53375       GPRINT.X = GPRINT.X+18 : GPRINT.Y = 178 : GPRINT.SET% = 2
53380       GPRINT.STR$ = FNLSD$(STR$(FIX(X1+SGN(X1)*.5)+I))
53385       GOSUB 36000 'call gprint(x, y, s, str)
53390     NEXT I
53395 '
53400 '....and now for the Y-vertical axis.
53405     FOR I = 0 TO SYS.X.PART STEP SY%
53410       GPRINT.X = 22 : GPRINT.Y = 200-(I/SYS.X.PART*168+24) : GPRINT.SET% = 1
53415       GPRINT.STR$ = FNLSD$(STR$(1/SYS.X.PART*I*Y.RANGE+Y1))
53420       GOSUB 36000 'call gprint(x, y, s, str)
53425     NEXT I
53430 '
53435 'Init X/Y vars, and read data points until we reach the data flags.
53440     WINDOW (-.12, -.122) - (1.1, 1.065)
53445     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
53450     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 53505
53455       NEWLINE = 1
53460       INPUT #1, X, Y
53465       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 53485
53470         ON NEWLINE GOSUB 53565, 53600
53475         INPUT #1, X, Y
53480       GOTO 53465
53485     GOTO 53450
53490 '
53495 'Process label entries. Each consists of a label coordinate (row, col)
53500 'plus a label.
53505     GOSUB 35000 'call label()
53510     FOR I = 1 TO NUM.LABELS%
53515       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)
53520       PRINT LABEL.STR$(I);
53525     NEXT I
```

```
53530 '
53535 'Close data file, put pen away, and return.
53540     CLOSE #1
53545     SYS.SCRN.CHANGED% = TRUE%
53550     RETURN
53555 '
53560 'Subroutine for beginning of line. Note the change to NEWLINE.
53565     X = (FNLOG10(X) - X1)/X.RANGE
53570     Y = (Y - Y1)/Y.RANGE
53575     PSET (X,Y)
53580     NEWLINE = 2
53585     RETURN
53590 '
53595 'Subroutine to simply plot the point.
53600     X = (FNLOG10(X) - X1)/X.RANGE
53605     Y = (Y - Y1)/Y.RANGE
53610     LINE -(X,Y)
53615     RETURN
```

AD-A159 808 GRAPS: GRAPHICAL PLOTTING SYSTEM(U) NAVAL OCEAN SYSTEMS 2/2
CENTER SAN DIEGO CA R T LAIRD JUL 85 NOSC/TD-820

UNCLASSIFIED F/G 9/2 NL





POLARV MODULE

```
54000 REM
54005 REM ************************************************************************
54010 REM *
54015 REM *  Routine Name:     polarv
54020 REM *
54025 REM *  Description   :     This routine displays the graph given
54030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
54035 REM *                  The graph is displayed in the polar co-
54040 REM *                  ordinate system.
54045 REM *
54050 REM *  To Call       :     GOSUB 54000
54055 REM *
54060 REM *  Globals       :     The following variable(s) are affected
54065 REM *                  by this routine:
54070 REM *
54075 REM *                  LABEL.pp - The parameters to the LABEL,
54080 REM *                       data file text/label extraction,
54085 REM *                       routine.
54090 REM *
54095 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
54100 REM *                       whether or not the screen has
54105 REM *                       changed (switched).
54110 REM *
54115 REM *  Edit History  :     1) Robin Laird 4/4/85
54120 REM *
54125 REM ************************************************************************
54130 REM
54135 '
54140 'Re-open data file for input as unit #1.
54145     OPEN SYS.FILENAME$ FOR INPUT AS #1
54150 '
54155 'Draw graph boundaries (graph border).
54160     SCREEN 2,0,0,0
54165     WINDOW (-1.3,-1.05) - (1.3,1.05)
54170 '
54175 'Draw polar grid. First, 5 concentric circles from radii 0.2 to 1.0.
54180     FOR R = .195 TO .975 STEP .195
54185       CIRCLE (0,0), R
54190     NEXT R
54195 '
54200 'Now for the "cross-hairs" (the grid axes).
54205     LINE (-.975,0) - (.975,0)
54210     LINE (0,1.05) - (0,-1.04)
54215 '
54220 'Draw the grid labels (these will be the same for all graphs).
54225     P = 0
54230     FOR I = -40 TO 10 STEP 10
54235       IF I > 0 THEN S$ = "+" ELSE IF I = 0 THEN S$ = "  " ELSE S$ = ""
54240       GPRINT.X = 290 + P*8 : GPRINT.Y = 102 : GPRINT.SET% = 1
54245       GPRINT.STR$ = S$ + FNLSD$(STR$(I))
54250       GOSUB 36000 'call gprint(x, y, s, str)
54255       P = P + 6
```

```
54260     NEXT I
54265     LOCATE 15, 37 + P - 6
54270     PRINT "DBI";
54275 '
54280 'Init A/R vars, and read data points until we reach the data flags.
54285     WINDOW (-1.3,-1.05) - (1.3,1.05)
54290     A = NOT SYS.FLAG.VALUE : R = NOT SYS.FLAG.VALUE
54295     IF A = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 54350
54300       NEWLINE = 1
54305       INPUT #1, A, R
54310       IF R = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 54330
54315         ON NEWLINE GOSUB 54410, 54445
54320         INPUT #1, A, R
54325       GOTO 54310
54330     GOTO 54295
54335 '
54340 'Process label entries. Each consists of a label coordinate (row, col)
54345 'plus a label.
54350     GOSUB 35000 'call label()
54355     FOR I = 1 TO NUM.LABELS%
54360       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)+10
54365       PRINT LABEL.STR$(I);
54370     NEXT I
54375 '
54380 'Close data file, put pen away, and return.
54385     CLOSE #1
54390     SYS.SCRN.CHANGED% = TRUE%
54395     RETURN
54400 '
54405 'Subroutine for beginning of line. Note the change to NEWLINE.
54410     A = A*PI/180
54415     IF R < -40 THEN R = 0 ELSE R = (R+40)/50
54420     PSET (R*SIN(A),R*COS(A))
54425     NEWLINE = 2
54430     RETURN
54435 '
54440 'Subroutine to simply plot the point.
54445     A = A*PI/180
54450     IF R < -40 THEN R = 0 ELSE R = (R+40)/50
54455     LINE -(R*SIN(A),R*COS(A))
54460     RETURN
```

SMITHV MODULE

```
55000 REM
55005 REM ************************************************************************
55010 REM *
55015 REM *  Routine Name:     smithv
55020 REM *
55025 REM *  Description   :     This routine displays the graph given
55030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
55035 REM *                  The graph displays X and Y using the
55040 REM *                  smith coordinate system.
55045 REM *
55050 REM *  To Call       :     GOSUB 55000
55055 REM *
55060 REM *  Globals       :     The following variable(s) are affected
55065 REM *                  by this routine:
55070 REM *
55075 REM *                  LABEL.pp - The parameters to the LABEL,
55080 REM *                       data file text/label extraction,
55085 REM *                       routine.
55090 REM *
55095 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
55100 REM *                       whether or not the screen has
55105 REM *                       changed (switched).
55110 REM *
55115 REM *  Edit History  :     1) Robin Laird 4/16/85
55120 REM *
55125 REM ************************************************************************
55130 REM
55135 '
55140 'Re-open data file for input as unit #1.
55145     OPEN SYS.FILENAME$ FOR INPUT AS #1
55150 '
55155 'Setup screen and window aspects of display.
55160     SCREEN 2,0,0,0
55165     WINDOW (-1.3,-1.05) - (1.3,1.05)
55170 '
55175 'Draw circular axes (at values 0, .3, 1, 3).
55180     X = 0 : Y = -.75 : R = .24
55185     FOR C = 1 TO 4
55190       CIRCLE (X, Y), R
55195       Y = Y + .255 : R = R + .24
55200     NEXT C
55205 '
55210 'Draw semi-circle axes (at values -1, -.5, 0, .5, 1)....
55215     X = -1 : Y = -1 : R = 1
55220     FOR A = 1 TO 2
55225       CIRCLE (X, Y),R,,0,3.1415927#/2-(R^4*.0426)
55230       X = X - 1 : R = R + 1
55235     NEXT A
55240 '
55245 '... and now for the other side....
55250     X = 1 : Y = -1 : R = 1
55255     FOR A = 1 TO 2
```

```
55260       CIRCLE (X, Y),R,,3.1415927#/2+(R^4*.0426),3.1415927#
55265       X = X + 1 : R = R + 1
55270     NEXT A
55275 '
55280 '... and finally the bar down the middle.
55285     LINE (0,-1) - (0,1.1)
55290 '
55295 'Label the reactance circles.
55300     GPRINT.X = 93 : GPRINT.Y = 96 : GPRINT.SET% = 1
55305     GPRINT.STR$ = "-1"
55310     GOSUB 36000 'call gprint(x, y, s, str)
55315 '
55320     GPRINT.X = 132 : GPRINT.Y = 42 : GPRINT.SET% = 1
55325     GPRINT.STR$ = "-.5"
55330     GOSUB 36000 'call gprint(x, y, s, str)
55335 '
55340     GPRINT.X = 308 : GPRINT.Y = 6 : GPRINT.SET% = 1
55345     GPRINT.STR$ = "0"
55350     GOSUB 36000 'call gprint(x, y, s, str)
55355 '
55360     GPRINT.X = 486 : GPRINT.Y = 42 : GPRINT.SET% = 1
55365     GPRINT.STR$ = ".5"
55370     GOSUB 36000 'call gprint(x, y, s, str)
55375 '
55380     GPRINT.X = 544 : GPRINT.Y = 96 : GPRINT.SET% = 1
55385     GPRINT.STR$ = "1"
55390     GOSUB 36000 'call gprint(x, y, s, str)
55395 '
55400 'Init K/I vars, and read data points until we reach the data flags.
55405     WINDOW (-1.3,-1.05) - (1.3,1.05)
55410     K = NOT SYS.FLAG.VALUE : I = NOT SYS.FLAG.VALUE
55415     IF K = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 55470
55420       NEWLINE = 1
55425       INPUT #1, K, I
55430       IF I = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 55450
55435         ON NEWLINE GOSUB 55530, 55555
55440         INPUT #1, K, I
55445       GOTO 55430
55450     GOTO 55415
55455 '
55460 'Process label entries. Each consists of a label coordinate (row, col)
55465 'plus a label.
55470     GOSUB 35000 'call label()
55475     FOR I = 1 TO NUM.LABELS%
55480       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)+10
55485       PRINT LABEL.STR$(I);
55490     NEXT I
55495 '
55500 'Close data file, put pen away, and return.
55505     CLOSE #1
55510     SYS.SCRN.CHANGED% = TRUE%
55515     RETURN
55520 '
55525 'Subroutine for beginning of line. Note the change to NEWLINE.
```

```
55530     PSET (FNCVTYSS(K,I), FNCVTXSS(K,I))
55535     NEWLINE = 2
55540     RETURN
55545 '
55550 'Subroutine to simply plot the point.
55555     LINE -(FNCVTYSS(K,I), FNCVTXSS(K,I))
55560     RETURN
```

LOGLOGV MODULE

```
56000 REM
56005 REM ************************************************************************
56010 REM *
56015 REM *  Routine Name:     loglogv
56020 REM *
56025 REM *  Description   :     This routine displays the graph given
56030 REM *                  by SYS.FILENAME$ and SYS.GRAPHTYPE$.
56035 REM *                  The graph displays both X and Y
56040 REM *                  logarithmically along their respective
56045 REM *                  axes. Note that the small scale numbers
56050 REM *                  that appear on the top and left axes
56055 REM *                  of the plot are not included in the
56060 REM *                  displayed version.
56065 REM *
56070 REM *  To Call       :     GOSUB 56000
56075 REM *
56080 REM *  Globals       :     The following variable(s) are affected
56085 REM *                  by this routine:
56090 REM *
56095 REM *                  LABEL.pp - The parameters to the LABEL,
56100 REM *                       data file text/label extraction,
56105 REM *                       routine.
56110 REM *
56115 REM *                  SYS.SCRN.CHANGED% - Boolean indicating
56120 REM *                       whether or not the screen has
56125 REM *                       changed (switched).
56130 REM *
56135 REM *  Edit History  :     1) Robin Laird 5/6/85
56140 REM *
56145 REM ************************************************************************
56150 REM
56155 '
56160 'Re-open data file as unit #1.
56165     OPEN SYS.FILENAME$ FOR INPUT AS #1
56170 '
56175 'Read X min/max and Y min/max, and calculate X and Y range and num cycles.
56180     INPUT #1, X1, X2, Y1, Y2
56185     X1 = FNLOG10(X1) : X2 = FNLOG10(X2)
56190     Y1 = FNLOG10(Y1) : Y2 = FNLOG10(Y2)
56195     X.RANGE = X2 - X1
56200     Y.RANGE = Y2 - Y1
56205     Z1% = ABS(X2 - X1) : IF Z1% = 0 THEN Z1% = 1
56210     Z2% = ABS(Y2 - Y1) : IF Z2% = 0 THEN Z2% = 1
56215 '
56220 'Calculate steps for FOR-NEXT loops (how many sets of tic marks).
56225     SX% = Z1%18 : IF Z1% MOD 18 <> 0 THEN SX% = SX% + 1
56230     SY% = Z2%16 : IF Z2% MOD 16 <> 0 THEN SY% = SY% + 1
56235 '
56240 'Draw graph boundaries (graph border).
56245     SCREEN 2,0,0,0
56250     WINDOW (-.12, -.122) - (1.1, 1.065)
56255     LINE (0, 0) - (1, 1),,B
```

```
56260 '
56265 'Draw axes tic marks, first for the X-horizontal axis. Note that this
56270 'is done for both the upper/lower sides of the border.
56275     FOR X = 0 TO 1
56280       FOR C = 1 TO Z1% STEP SX%
56285         FOR Y = 2 TO 10
56290           IF (Y=10 AND C=Z1%) THEN P=-1 ELSE P=FNLOG10(Y * 10^(C-1))/Z1%
56295           LINE (P, X-8.000001E-03) - (P, X+8.000001E-03)
56300         NEXT Y
56305       NEXT C
56310     NEXT X
56315 '
56320 '....do the same for the Y-vertical axis.
56325     FOR Y = 0 TO 1
56330       FOR C = 1 TO Z2% STEP SY%
56335         FOR X = 2 TO 10
56340           IF (X=10 AND C=Z2%) THEN P=-1 ELSE P=FNLOG10(X * 10^(C-1))/Z2%
56345           LINE (Y-.006, P) - (Y+.006, P)
56350         NEXT X
56355       NEXT C
56360     NEXT Y
56365 '
56370 'Draw scale (tic) numbers, first for the X-horizontal, bottom axis.
56375     FOR I = 0 TO Z1% STEP SX%
56380       GPRINT.X = I/Z1%*520+60 : GPRINT.Y = 184 : GPRINT.SET% = 1%
56385       GPRINT.STR$ = "10"
56390       GOSUB 36000 'call gprint(x, y, s, str)
56395       GPRINT.X = GPRINT.X+18 : GPRINT.Y = 178 : GPRINT.SET% = 2%
56400       GPRINT.STR$ = FNLSD$(STR$(FIX(X1+SGN(X1)*.5)+I))
56405       GOSUB 36000 'call gprint(x, y, s, str)
56410     NEXT I
56415 '
56420 '....and now for the Y-vertical axis.
56425     FOR I = 0 TO Z2% STEP SY%
56430       GPRINT.X = 24 : GPRINT.Y = 200-(I/Z2%*168+20) : GPRINT.SET% = 1%
56435       GPRINT.STR$ = "10"
56440       GOSUB 36000 'call gprint(x, y, s, str)
56445       GPRINT.X = 42 : GPRINT.Y = GPRINT.Y-6 : GPRINT.SET% = 2%
56450       GPRINT.STR$ = FNLSD$(STR$(FIX(Y1+SGN(Y1)*.5)+I))
56455       GOSUB 36000 'call gprint(x, y, s, str)
56460     NEXT I
56465 '
56470 'Init X/Y vars, and read data points until we reach the data flags.
56475     WINDOW (-.12, -.122) - (1.1, 1.065)
56480     X = NOT SYS.FLAG.VALUE : Y = NOT SYS.FLAG.VALUE
56485     IF X = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 56540
56490       NEWLINE = 1
56495       INPUT #1, X, Y
56500       IF Y = SYS.FLAG.VALUE OR EOF(1) THEN GOTO 56520
56505         ON NEWLINE GOSUB 56600, 56635
56510         INPUT #1, X, Y
56515       GOTO 56500
56520     GOTO 56485
56525 '
```

```
56530 'Process label entries. Each consists of a label coordinate (row, col)
56535 'plus a label.
56540     GOSUB 35000 'call label()
56545     FOR I = 1 TO NUM.LABELS%
56550       LOCATE LABEL.POS(1, I), LABEL.POS(2, I)
56555       PRINT LABEL.STR$(I);
56560     NEXT I
56565 '
56570 'Close data file, put pen away, and return.
56575     CLOSE #1
56580     SYS.SCRN.CHANGED% = TRUE%
56585     RETURN
56590 '
56595 'Subroutine for beginning of line. Note the change to NEWLINE.
56600     X = (FNLOG10(X) - X1)/X.RANGE
56605     Y = (FNLOG10(Y) - Y1)/Y.RANGE
56610     PSET (X,Y)
56615     NEWLINE = 2
56620     RETURN
56625 '
56630 'Subroutine to simply plot the point.
56635     X = (FNLOG10(X) - X1)/X.RANGE
56640     Y = (FNLOG10(Y) - Y1)/Y.RANGE
56645     LINE -(X,Y)
56650     RETURN
```

APPENDIX B
ORIGINAL PLOTTING PROGRAM LISTINGS

The listings that follow are the original plotting programs. The plotting routines implemented in GRAPS were derived from these programs. I have, however, tried to improve on the code so that it is now more readable and easier to "follow". The five (5) programs included here represent routines for plotting the *linear, bilinear, loglinear, polar,* and *Smith* graph types. The horizontal versions of the *loglinear* and the *loglog* graph types have no predecessors.

Note that the **LOGLINEAR** program includes routines for plotting three graph types: *linear, bilinear,* and *loglinear.* The other programs plot only the one graph type specified in the program name.

LINEAR PLOT PROGRAM

```
100 REM  LINPLT.BAS  (CONVERT FROM LINPLOT.FOR BY TINNA MOK)
120 REM  THIS PROGRAM MAKES LINEAR PLOTS FORM DATA IN FILE ON UNIT 1
140 REM  MODIFIED BY J. LOGAN FROM PROGRAM BY JOHN YEN - 9/13/84
160 ON ERROR GOTO 4280
180 DIMENSION USTR(6)
200 DIMENSION W(4),Z(4),LP(5),UX(6),VY(6),TTX(4),TTY(4)
220 W(1)=0.
240 W(2)=1.
260 W(3)=0.
280 W(4)=1.
300 Z(1)=8000.
320 Z(2)=1000.
340 Z(3)=6500.
360 Z(4)=1000.
380 LP(1)=3
400 LP(2)=6
420 LP(3)=4
440 LP(4)=2
460 LP(5)=5
480 VLT$(1)="; "
500 VLT$(2)="6;"
520 VLT$(3)="2;"
540 VLT$(4)="4;"
560 VLT$(5)="1;"
580 PRINT "INPUT DATA FILENAME (XXXXXXX.DAT)PLEASE";
600 INPUT F$
620 OPEN F$ FOR INPUT AS FILE #1
640 REM
660 REM READ RANGE FROM UNIT 1
680 REM
700 INPUT #1, X0,X1,Y0,Y1
720 IF (ITT < 1) THEN ITT=1
740 IF (ITT > 4) THEN ITT=1
760 XL=X1-X0
780 YL=Y1-Y0
800 REM
820 REM
840 REM  AXES LABELS
860 USTR(1)=0.
880 USTR(2)=.2
900 USTR(3)=.4
920 USTR(4)=.6
940 USTR(5)=.8
960 USTR(6)=1.
980 REM  X-AXIS LABEL COORDINATES
1000 UX(1)=800.
1020 UX(2)=2200.
1040 UX(3)=3800.
1060 UX(4)=5400.
1080 UX(5)=7000.
1100 UX(6)=8650.
1120 UY=800.
```

```
1140 REM  Y-AXIS LABEL COORDINATES
1160 VX=400.
1180 VY(1)=1000.
1200 VY(2)=2250.
1220 VY(3)=3550.
1240 VY(4)=4850.
1260 VY(5)=6150.
1280 VY(6)=7450.
1300 REM
1320 REM  NORMALIZATION LABELS
1340 TTX(1)=1200.
1360 TTX(2)=1200.
1380 TTX(3)=5500.
1400 TTX(4)=5500.
1420 TTY(1)=2000.
1440 TTY(2)=7200.
1460 TTY(3)=7200.
1480 TTY(4)=2000.
1500 REM  INITIALIZATION : SI; FOR CORRECT LETTER SIZE
1520 REM SP1; FOR (BLACK) PEN
1540 REM LT; FOR CONTINUOUS LINE
1560 REM DT ; SETS " " AS END OF LABEL STRINGS
1580 REM IP...; SET PLOT BOUNDARIES (OVERRODE LATER)
1600 PRINT " IN;IP1000,1000,9000,7500;SI;SP1;LT;DT "
1620 REM
1640 REM ------------------------------------------------
1660 REM  DRAW BOUNDARY LINES
1680 X=0.
1700 Y=0.
1720 GOSUB 4460
1740 PX=PX
1760 PY=PY
1780 PRINT " PUPA";PX;",";PY;";PD;"
1800 FOR I = 1 TO 4
1820 IF I=1 THEN Y=1
1840 IF I=2 THEN X=1
1860 IF I=3 THEN Y=0
1880 IF I=4 THEN X=0
1900 GOSUB 4460
1920 PX=PX
1940 PY=PY
1960 PRINT " PA";PX;",";PY;";"
1980 NEXT I
2000 REM ------------------------------------------------
2020 REM
2040 L=1
2060 RX=0.
2080 RY=0.
2100 REM  READ BOUNDARIES FROM UNIT 1
2120 REM    ITT IS POSITION OF NORMALIZATION PRINTOUT [DEFAULT 1]
2140 REM     _____________
2160 REM    |2   3        |
2180 REM    |             |
2200 REM    |1   4        |
```

```
2220 REM    |_____|
2240 REM    X0,Y0 ARE THE LOWER BOUNDS
2260 REM    X1,Y1 ARE THE UPPER BOUNDS
2280 REM ------------------------------------------------
2300 XJ=0.1
2320 YJ=0.2
2340 REM  PRINT AXES TICKS
2360 FOR I=1 TO 2
2380 FOR NX=1 TO 9
2400 XN=NX*XJ
2420 X=XN
2440 IF I = 1 THEN Y=0.
2460 IF I = 2 THEN Y=1.
2480 GOSUB 4460
2500 PX=PX
2520 PY=PY
2540 PRINT " PUPA";PX;",";PY;";XT;"
2560 NEXT NX
2580 NEXT I
2600 REM  XT; IS FOR TICKS ON X-AXIS
2620 REM ----------------------------
2640 FOR I=1 TO 2
2660 FOR NY=1 TO 4
2680 YN=NY*YJ
2700 IF I=1 THEN X=0
2720 IF I=2 THEN X=1.
2740 Y=YN
2760 GOSUB 4460
2780 PX=PX
2800 PY=PY
2820 PRINT " PUPA";PX;",";PY;";YT;"
2840 NEXT NY
2860 NEXT I
2880 REM ------------------------------------------------
2900 REM  YT; IS FOR TICKS ON Y-AXIS
2920 REM
2940 REM  WRITE AXES LABELS
2960 REM     DI1,0; SET DIRECTION OF LABEL PRINTING
2980 FOR M=1 TO 6
3000 PRINT " DI1,0;PUPA";UX(M);",";UY
3020 TEMP1=USTR(M)*XL+X0
3040 PRINT " LB ";TEMP1;" "
3060 NEXT M
3080 FOR M=1 TO 6
3100 PRINT " DI1,0;PUPA";VX;",";VY(M)
3120 TEMP1=USTR(M)*YL+Y0
3140 PRINT " LB ";TEMP1;" "
3160 REM  NEEDS " " TO END LABEL STRING (SEE "DT" ABOVE)
3180 NEXT M
3200 REM ------------------------------------------------
3220 REM
3240 REM  LIMIT OF 5 DATA GROUPS
3260 REM  LAST GROUP ENDS WITH RX=-1.234
3280 IF (L > 5) THEN GOTO 3980
```

```
16000 K=1
16050 IF B$<>"LG" THEN GOTO 16200
16100 INPUT #1, AY,AX
16150 GOTO 16250
16200 INPUT #1, AX,AY
16250 LINPUT #1, ASTR$
16300 IF (B$="LG") THEN GOTO 16400 ELSE PRINT " DI1,0;PUPA";AX;",";AY
16350 GOTO 16450
16400 PRINT " DI0,-1;PUPA";AX;",";AY
16450 PRINT " LB",ASTR$;" "
16500 GOTO 16050
16550 REM --TEST FOR END OF FILE ERROR FLAG IN DATA FILE
16600 IF (ERR=11%) THEN RESUME 16650
16650 PRINT " PU;SP0;"
16700 CLOSE #1
16750 GOTO 17100
16800 REM
16850 REM --ROUTINE TO CONVERT NORMALIZED COORDINATES INTO PLOTTER COORDINATES
16900 REM --SUBROUTINE PEER(X,Y,W,Z,PX,PY)
16950 PX=Z(1)*(X-W(1))/(W(2)-W(1))+Z(2)
17000 PY=Z(3)*(Y-W(3))/(W(4)-W(3))+Z(4)
17050 RETURN
17100 END
```

```
13300 REM   3 : (VIOLET) _. _.
13350 REM   4 : (ORANGE) __. __.
13400 REM   5 : (BLUE)  . . . .
13450 PRINT " SI;SP";LP(L);";"
13500 PRINT " LT";VLT$(L)
13550 L=L+1
13600 I=0
13650 REM ------------------------------------------------
13700 REM --READ DATA CARDS
13750 REM --EACH GROUP OF DATA CARDS ENDS WITH RY=-1.234
13800 IF B$<>"LG" THEN GOTO 14000
13850 INPUT #1, RY,RX
13900 RRX=RY
13950 GOTO 14100
14000 INPUT #1, RX,RY
14050 RRX=RX
14100 IF (RY=-1.234 AND B$="LIN") THEN GOTO 12950
14150 IF (RY=-1.234 AND B$="LIN2") THEN GOTO 15350
14200 IF (RX=-1.234 AND B$="LG") THEN GOTO 12950
14250 IF (RX<=0. AND B$="LG") THEN GOTO 15050
14300 IF (B$<>"LG") THEN GOTO 14550
14350 RX=LOG10(RX)
14400 RY=1.-(RY-Y0)/YL
14450 GOTO 14600
14500 REM --NORMALIZE, SUCH THAT [X0,X1] AND [Y0,Y1] BECOME [0,1]
14550 RY=(RY-Y0)/YL
14600 RX=(RX-X0)/XL
14650 X=RX
14700 Y=RY
14750 GOSUB 16900
14800 RX=X
14850 RY=Y
14900 IF I <> 0 THEN GOTO 15150
14950 PRINT " PUPA";PX;",";PY;";PD;"
15000 GOTO 15200
15050 PRINT "DATA ERROR AT ";RY;",";RX;")"
15100 GOTO 16650
15150 PRINT " PA";PX;",";PY;";"
15200 I=I+1
15250 GOTO 13800
15300 IF B$="LIN" THEN GOTO 15600
15350 YL=YL2
15400 Y0=Y02
15450 GOTO 12950
15500 REM ------------------------------------------------
15550 REM --WRITES NORMALIZATION INFORMATION
15600 TX=TTX(ITT)
15650 TY=TTY(ITT)
15700 PRINT " SI;SP5;"
15750 TY=TY-200.
15800 TY=TY-300.
15850 TY=TY-200.
15900 REM ------------------------------------------------
15950 REM --READ END LABELS
```

```
10600 IX0=IX0+1
10650 NEXT M
10700 PRINT " SI"
10750 FOR M=1 TO 6
10800 PRINT " DI0,-1;PUPA";VX;",";VY(M)
10850 TEMP1=USTR(M)*YL+Y0
10900 PRINT " LB";TEMP1;" "
10950 NEXT M
11000 GOTO 12950
11050 REM ---NEEDS " " TO END LABEL STRING (SEE "DT" ABOVE)
11100 REM ------------------------------------------------
11150 FOR I=1 TO 2
11200 FOR NY=1 TO 4
11250 YN=NY*YJ
11300 Y=YN
11350 IF I=1 THEN X=0.
11400 IF I=2 THEN X=1.
11450 GOSUB 16900
11500 PRINT " PUPA";PX;",";PY;";YT;"
11550 NEXT NY
11600 NEXT I
11650 REM --YT; IS FOR TICKS ON Y-AXIS
11700 REM ------------------------------------------------
11750 REM
11800 REM --WRITE AXES LABELS
11850 REM --   DI1,0; SET DIRECTION OF LABEL PRINTING
11900 FOR M=1 TO 6
11950 PRINT " DI1,0;PUPA";UX(M);",";UY
12000 TEMP1=USTR(M)*XL+X0
12050 PRINT " LB";TEMP1;" "
12100 NEXT M
12150 FOR M = 1 TO 6
12200 PRINT " DI1,0;PUPA";VX;",";VY(M)
12250 TEMP1=USTR(M)*YL+Y0
12300 PRINT " LB";TEMP1;" "
12350 NEXT M
12400 IF B$="LIN" THEN GOTO 12950
12450 IF B$="LIN2" THEN GOTO 12500
12500 FOR M=1 TO 6
12550 PRINT " DI1,0;PUPA";VXX;",";VY(M)
12600 TEMP1=USTR(M)*YL2+Y02
12650 PRINT " LB";TEMP1;" "
12700 NEXT M
12750 REM --NEEDS " " TO END LABEL STRING (SEE "DT" ABOVE)
12800 REM ------------------------------------------------
12850 REM --LIMIT OF 5 DATA GROUPS
12900 REM --LAST GROUP ENDS WITH RX=-1.234
12950 IF (L>5 AND B$="LG") THEN  GOTO 16000
13000 IF (RRX=-1.234 AND B$="LG") THEN GOTO 16000
13050 IF (L>5 AND B$<>"LG") THEN GOTO 15600
13100 IF (RRX=-1.234 AND B$<>"LG") THEN GOTO 15600
13150 REM --SELECTS PEN COLOR AND LINE-TYPE
13200 REM   1 : (RED)   _______
13250 REM   2 : (GREEN) _____
```

```
7900 IC=IC+1
7950 J=NX*10**(I-1)
8000 IF I=JCYCLE AND NX=10 THEN GOTO 8300
8050 XN=10*LOG10(J)*XJ/JCYCLE
8100 X=XN
8150 Y=0.
8200 GOSUB 16900
8250 PRINT "PUPA";PX;",";PY;";XT;"
8300 UX(IC)=UXLAST
8350 NEXT NX
8400 NEXT I
8450 IC=1
8500 FOR I=1 TO JCYCLE
8550 FOR NX=2 TO 10
8600 IC=IC+1
8650 J=NX*10**(I-1)
8700 IF I=JCYCLE AND NX=10 THEN GOTO 9000
8750 XN=10*LOG10(J)*XJ/JCYCLE
8800 X=XN
8850 Y=1
8900 GOSUB 16900
8950 PRINT " PUPA";PX;",";PY;";XT;"
9000 UX(IC)=PX
9050 GOTO 9100
9100 NEXT NX
9150 NEXT I
9200 REM --END OF X-TICKS FOR LOG-LINEAR (XT; IS FOR TICKS ON X-AXIS)
9250 REM ------------------------------------------------
9300 FOR I=1 TO 2
9350 FOR NY=1 TO 9
9400 YN=NY*XJ
9450 IF I=1 THEN X=0.
9500 IF I=2 THEN X=1.
9550 Y=YN
9600 GOSUB 16900
9650 PRINT " PUPA";PX;",";PY;";YT;"
9700 NEXT NY
9750 NEXT I
9800 REM ---END OF Y-AXIS FOR LOG-LINEAR
9850 JC=0
9900 IX0=0
9950 FOR M=1 TO IC
10000 JC=JC+1
10050 IF (JC =10) THEN JC=1
10100 PRINT "DI0,-1;PUPA";UX(M);",";UY
10150 PRINT "SI.11,.16;LB";LABRH$(JC);" "
10200 IF (JC > 1) THEN GOTO 10650
10250 UXM=UX(M)-75.
10300 PRINT " SI"
10350 PRINT " DI0,-1;PUPA";UXM;",";UY1
10400 PRINT " LB";"10";" "
10450 PRINT " DI0,-1;PUPA";UX(M);",";UY2
10500 IXPO=FIX(X0)+IX0
10550 PRINT " SI.11,.16;LB";IXPO;" "
```

```
5200 GOTO 5500
5250 REM --INITIALIZATION : SI; FOR CORRECT LETTER SIZE
5300 REM            SP1; FOR (BLACK) PEN
5350 REM            LT; FOR CONTINUOUS LINE
5400 REM            DT ; SETS " " AS END OF LABEL STRINGS
5450 REM            IP...; SET PLOT BOUNDARIES (OVERRODE LATER)
5500 PRINT " IN;IP1000,1000,9000,7500;SI;SP1;LT;DT "
5550 REM
5600 REM ------------------------------------------------
5650 REM --DRAW BOUNDARY LINES
5700 X=0.
5750 Y=0.
5800 GOSUB 16900
5850 PRINT "PUPA";PX;",";PY;";PD;"
5900 FOR I=1 TO 4
5950 IF I=1 THEN Y=1.
6000 IF I=2 THEN X=1
6050 IF I=3 THEN Y=0
6100 IF I=4 THEN X=0
6150 GOSUB 16900
6200 PRINT " PA";PX;",";PY;";"
6250 NEXT I
6300 REM ------------------------------------------------
6350 L=1
6400 RX=0.
6450 RY=0.
6500 REM --READ BOUNDARIES
6550 REM   ITT IS POSITION OF NORMALIZATION PRINTOUT [DEFAULT 1]
6600 REM
6650 REM            |2          3|
6700 REM            |             |
6750 REM            |1          4|
6800 REM            |_____________|
6850 REM   X0,Y0 ARE THE LOWER BOUNDS
6900 REM   X1,Y1 ARE THE UPPER BOUNDS
6950 XJ=0.1
7000 YJ=0.2
7050 IF B$="LG" THEN GOTO 7750
7100 REM ------------------------------------------------
7150 REM --PRINT AXES TICKS FOR 2 LINEAR PLOTS
7200 FOR I=1 TO 2
7250 FOR NX=1 TO 9
7300 XN=NX*XJ
7350 X=XN
7400 IF I=1 THEN Y=0.
7450 IF I=2 THEN Y=1.
7500 GOSUB 16900
7550 PRINT " PUPA";PX;",";PY;";XT;"
7600 NEXT NX
7650 NEXT I
7700 GOTO 11150
7750 IC=1
7800 FOR I=1 TO JCYCLE
7850 FOR NX=2 TO 10
```

```
2500 GOTO 2800
2550 INPUT #1,Y0,Y1,X0,X1
2600 X0=LOG10(X0)
2650 X1=LOG10(X1)
2700 JCYCLE=ABS(FIX(X1-X0))
2750 PRINT "NO. OF CYCLE= ";JCYCLE
2800 XL=X1-X0
2850 YL=Y1-Y0
2900 REM
2950 REM --AXES LABELS
3000 READ USTR(1),USTR(2),USTR(3),USTR(4),USTR(5),USTR(6)
3050 IF B$<>"LG" THEN 3900
3100 READ LABRH$(1),LABRH$(2),LABRH$(3),LABRH$(4),LABRH$(5),LABRH$(6),LABRH$(7)
3150 READ LABRH$(8),LABRH$(9)
3200 REM --X-AXIS LABEL COORDINATES
3250 UX(1)=1000.
3300 UXLAST=9000.
3350 UY=800.
3400 UY1=7850.
3450 UY2=7700.
3500 VX=800.
3550 VY(1)=7650.
3600 VY(2)=6350.
3650 VY(3)=5050.
3700 VY(4)=3750.
3750 VY(5)=2450.
3800 VY(6)=1150.
3850 GOTO 5500
3900 UX(1)=800.
3950 UX(2)=2400.
4000 UX(3)=4000.
4050 UX(4)=5600.
4100 UX(5)=7200.
4150 UX(6)=8850.
4200 UY=800
4250 REM --Y-AXIS LABEL COORDINATES
4300 VX=400.
4350 VXX=9100.
4400 VY(1)=1000.
4450 VY(2)=2250.
4500 VY(3)=3550.
4550 VY(4)=4850.
4600 VY(5)=6150.
4650 VY(6)=7450.
4700 REM
4750 REM --NORMALIZATION LABELS
4800 TTX(1)=1200.
4850 TTX(2)=1200.
4900 TTX(3)=5500.
4950 TTX(4)=5500.
5000 TTY(1)=2000.
5050 TTY(2)=7200.
5100 TTY(3)=7200.
5150 TTY(4)=2000.
```

LOGLINEAR PLOT PROGRAM

```
100 REM --LGLIN2.BAS
110 REM --THIS PROGRAM MAKES 3 KINDS OF PLOT FORMS.
120 !       1= LINEAR PLOTS : ONLY 2 CURVES CAN BE PLOTTED WHEN 'LIN2' IS CHOSEN
130 !         FIRST SET OF DATA USES LEFT HAND AXES, SECOND SET USES RIGHT
140 !       2= LINEAR PLOTS : MAX. 5 CURVES CAN BE PLOTTED WHEN 'LIN' IS CHOSEN
150 !         ONLY 1 LEFT HAND AXES IS LABELED
160 !       3= LOG-LINEAR PLOT : GIVE LOG-LINEAR PLOTS WHEN 'LG' IS SPECIFIED
170 !         BOTH LEFT & RIGHT HAND AXES ARE LABELED
180 REM --MODIFIED BY J. LOGAN FROM PROGRAM BY JOHN YEN - 9/13/84
190 REM --CONCATENATED AND CONVERTED BY TINNA MOK FROM 3 FORTRANS
200 !         TO 1 BASIC PROGRAM
210  REM --TEST THE END OF FILE AND ERROR FLAG IN DATA FILE
220 DATA 0.,1.,0.,1.
230 DATA 8000.,1000.,6500.,1000.
240 DATA 3,6,4,2,5
250 DATA "; ","6;","2;","4;","1;"
300 DATA 0.,.2,.4,.6,.8,1.
350 DATA "1","2","3","4","5","6","7"
400 DATA "8","9"
1050 ON ERROR GOTO 16600
1100 DIMENSION USTR(6)
1150 DIMENSION W(4),Z(4),LP(5),UX(50),VY(6),TTX(4),TTY(4)
1200 READ W(1),W(2),W(3),W(4)
1250 READ Z(1),Z(2),Z(3),Z(4)
1300 READ LP(1),LP(2),LP(3),LP(4),LP(5)
1350 READ VLT$(1),VLT$(2),VLT$(3),VLT$(4),VLT$(5)
1400 REM
1450 REM  --READ RANGE
1500 REM
1550 PRINT "INPUT DATA FILENAME (XXXXXXX.DAT) PLEASE";
1600 INPUT F$
1650 OPEN F$ FOR INPUT AS FILE #1
1700 PRINT
1750 PRINT "SELECT ONE OF THE FOLLOWING PLOTTING FORMAT:"
1755 PRINT
1760 PRINT "    LIN = 5 Linear plots with only left hand axis are labeled"
1800 PRINT "    LIN2= 2 Linear plots with left & right hand axis are labeled"
1850 PRINT "    LG  = Log-Linear plots with left & right hand axis are labeled"
1855 PRINT
1860 PRINT " WHICH ONE";
1900 INPUT B$
1950 PRINT
2000 IF B$="LIN" THEN GOTO 2350
2050 IF B$="LIN2" THEN GOTO 2200
2100 IF B$="LG" THEN GOTO 2550 ELSE PRINT "*****HAS TO BE LIN, LG OR LIN2*****"
2150 GOTO 1700
2200 INPUT #1, X0,X1,Y0,Y1,Y02,Y12
2250 YL2=Y12-Y02
2300 GOTO 2800
2350 INPUT #1, X0,X1,Y0,Y1
2400 IF (ITT<1) THEN ITT=1
2450 IF (ITT>4) THEN ITT=1
```

```
4660 REM --NORMALIZE, SUCH THAT [X0,X1] AND [Y0,Y1] BECOME [0,1]
4680 RX=(RX-X0)/XL
4700 RY=(RY-Y0)/YL
4720 REM --CALL PEER(RX,RY,W,Z,PX,PY)
4722 X=RX
4724 Y=RY
4730 GOSUB 5900
4732 PX=PX
4734 PY=PY
4740 IF I <> 0 THEN GOTO 4820
4760 PRINT " PUPA";PX;",";PY;";PD:"
4800 GOTO 4840
4820 PRINT " PA";PX;",";PY;";"
4840 I=I+1
4860 GOTO 4600
4880 YL=YL2
5000 Y0=Y02
5020 GOTO 4260
5040 REM
5060 REM
5080 REM --WRITES NORMALIZATION INFORMATION
5100 TX=TTX(ITT)
5120 TY=TTY(ITT)
5140 PRINT " SI;SP5;"
5260 TY=TY-200.
5340 TY=TY-300.
5420 TY=TY-200.
5500 REM
5520 REM --READ END LABELS FROM UNIT 1
5540 K=1
5580 INPUT #1, AX,AY
5590 IF AX=-9999 AND AY=-9999 THEN GOTO 5760
5600 INPUT #1, ASTR$
5620 IF ASTR$="-9999" THEN GOTO 5760
5662 PRINT " DI1,0;PUPA";AX;",";AY
5680 PRINT " LB";ASTR$;" "
5720 GOTO 5580
5760 CLOSE #1
5770 PRINT " PU;SP"
5800 GOTO 6000
5840 REM
5860 REM
5880 REM --ROUTINE TO CONVERT NORMALIZED COORDINATES INTO PLOTTER COORDINATES
5900 REM --SUBROUTINE PEER(X,Y,W,Z,PX,PY)
5940 PX=Z(1)*(X-W(1))/(W(2)-W(1))+Z(2)
5960 PY=Z(3)*(Y-W(3))/(W(4)-W(3))+Z(4)
5980 RETURN
6000 END
```

```
3704 FOR NY=1 TO 4
3706 YN=NY*YJ
3720 REM --CALL PEER(1.,YN,W,Z,PX,PY)
3722 X=1.
3724 Y=YN
3730 GOSUB 5900
3732 PX=PX
3734 PY=PY
3740 PRINT " PUPA";PX;",";PY;";YT;"
3760 REM --YT; IS FOR TICKS ON Y-AXIS
3800 NEXT NY
3810 REM ---------------------------------------------------------------------
3820 REM
3840 REM --WRITE AXES LABELS
3860 REM --   DI1,0; SET DIRECTION OF LABEL PRINTING
3880 FOR M=1 TO 6
3885 PRINT " DI1,0;PUPA";UX(M);",";UY
3900 TEMP1=USTR(M)*XL+X0
3905 PRINT " LB";TEMP1;" "
3910 NEXT M
3920 FOR M = 1 TO 6
3925 PRINT " DI1,0;PUPA";VX;",";VY(M)
3930 TEMP1=USTR(M)*YL+Y0
3935 PRINT " LB";TEMP1;" "
3940 NEXT M
3950 FOR M=1 TO 6
3955 PRINT " DI1,0;PUPA";VXX;",";VY(M)
3960 TEMP1=USTR(M)*YL2+Y02
3965 PRINT " LB";TEMP1;" "
3970 NEXT M
3980 REM --NEEDS " " TO END LABEL STRING (SEE "DT" ABOVE)
4180 REM
4200 REM
4220 REM --LIMIT OF 5 DATA GROUPS
4240 REM --LAST GROUP ENDS WITH RX=-1.234
4260 IF L>5 THEN  GOTO 5100
4280 IF RRX=-1.234 THEN GOTO 5100
4300 REM --SELECTS PEN COLOR AND LINE-TYPE
4320 REM   1 : (RED)    ________
4340 REM   2 : (GREEN) _____
4360 REM   3 : (VIOLET) _. _.
4380 REM   4 : (ORANGE) _ _.
4400 REM   5 : (BLUE)   . . . .
4420 PRINT " SI;SP";LP(L);";"
4460 PRINT " LT";VLT$(L)
4500 L=L+1
4520 I=0
4540 REM
4560 REM --READ DATA CARDS FROM UNIT 1
4580 REM --EACH GROUP OF DATA CARDS ENDS WITH RY=-1.234
4600 INPUT #1, RX,RY
4610 IF RX=-9999 AND RY=-9999 THEN GOTO 5760
4620 RRX=RX
4640 IF RY = -1.234 THEN GOTO 4880
```

```
3080 REM --CALL PEER(X,Y,W,Z,PX,PY)
3090 GOSUB 5900
3092 PX=PX
3094 PY=PY
3100 PRINT " PA";PX;",";PY;";"
3120 REM
3140 REM
3160 L=1
3180 RX=0.
3200 RY=0.
3220 REM --READ BOUNDARIES FROM UNIT 1
3240 REM   ITT IS POSITION OF NORMALIZATION PRINTOUT [DEFAULT 1]
3260 REM
3280 REM          |2          3|
3300 REM          |             |
3320 REM          |1          4|
3340 REM          |_____________|
3360 REM   X0,Y0 ARE THE LOWER BOUNDS
3380 REM   X1,Y1 ARE THE UPPER BOUNDS
3400 XJ=0.1
3420 YJ=0.2
3430 REM ------------------------------------------------------------------
3440 REM --PRINT AXES TICKS
3460 FOR NX=1 TO 9
3480 XN=NX*XJ
3500 REM --CALL PEER(XN,0.,W,Z,PX,PY)
3502 X=XN
3504 Y=0.
3510 GOSUB 5900
3512 PX=PX
3514 PY=PY
3520 PRINT " PUPA";PX;",";PY;";XT;"
3522 NEXT NX
3524 FOR NX=1 TO 9
3526 XN=NX*XJ
3540 REM --CALL PEER(XN,1.,W,Z,PX,PY)
3542 X=XN
3544 Y=1.
3550 GOSUB 5900
3552 PX=PX
3554 PY=PY
3560 PRINT " PUPA";PX;",";PY;";XT;"
3580 REM --XT; IS FOR TICKS ON X-AXIS
3620 NEXT NX
3640 FOR NY=1 TO 4
3660 YN=NY*YJ
3680 REM --CALL PEER(0.,YN,W,Z,PX,PY)
3682 X=0.
3684 Y=YN
3690 GOSUB 5900
3692 PX=PX
3694 PY=PY
3700 PRINT " PUPA";PX;",";PY;";YT;"
3702 NEXT NY
```

```
2200 REM --Y-AXIS LABEL COORDINATES
2220 VX=500.
2240 VXX=9100.
2260 VY(1)=1000.
2280 VY(2)=2250.
2300 VY(3)=3550.
2320 VY(4)=4850.
2340 VY(5)=6150.
2360 VY(6)=7450.
2380 REM
2400 REM --NORMALIZATION LABELS
2420 TTX(1)=1200.
2440 TTX(2)=1200.
2460 TTX(3)=5500.
2480 TTX(4)=5500.
2500 TTY(1)=2000.
2520 TTY(2)=7200.
2540 TTY(3)=7200.
2560 TTY(4)=2000.
2580 REM --INITIALIZATION : SI; FOR CORRECT LETTER SIZE
2600 REM              SP1; FOR (BLACK) PEN
2620 REM              LT; FOR CONTINUOUS LINE
2640 REM              DT ; SETS " " AS END OF LABEL STRINGS
2660 REM              IP...; SET PLOT BOUNDARIES (OVERRODE LATER)
2680 PRINT " IN;IP1000,1000,9000,7500;SI;SP1;LT;DT "
2720 REM
2740 REM
2760 REM --DRAW BOUNDARY LINES
2780 X=0.
2800 Y=0.
2820 REM CALL PEER(X,Y,W,Z,PX,PY)
2830 GOSUB 5900
2832 PX=PX
2834 PY=PY
2840 PRINT "PUPA";PX;",";PY;";PD;"
2880 Y=1.
2900 REM --CALL PEER(X,Y,W,Z,PX,PY)
2910 GOSUB 5900
2912 PX=PX
2914 PY=PY
2920 PRINT " PA";PX;",";PY;";"
2960 X=1.
2980 REM --CALL PEER(X,Y,W,Z,PX,PY)
2990 GOSUB 5900
2992 PX=PX
2994 PY=PY
3000 PRINT " PA";PX;",";PY;";"
3020 Y=0.
3040 REM --CALL PEER(X,Y,W,Z,PX,PY)
3050 GOSUB 5900
3052 PX=PX
3054 PY=PY
3060 PRINT " PA";PX;",";PY;";"
3070 X=0.
```

BILINEAR PLOT PROGRAM

```
1000 REM --LIN2PLOT.FOR
1220 REM --THIS PROGRAM MAKES LINEAR PLOTS FORM DATA IN FILE ON UNIT 1
1240 REM --ONLY 2 CURVES CAN BE PLOTTED
1260 REM --FIRST SET OF DATA USES LEFT HAND AXIS
1300 REM --SECOND SET USES RIGHT
1320 REM --MODIFIED BY J. LOGAN FROM PROGRAM BY JOHN YEN - 9/13/84
1380 DIMENSION USTR(6)
1400 DIMENSION W(4),Z(4),LP(5),UX(6),VY(6),TTX(4),TTY(4)
1420 W(1)=0.
1440 W(2)=1.
1460 W(3)=0.
1480 W(4)=1.
1500 Z(1)=8000.
1520 Z(2)=1000.
1540 Z(3)=6500.
1560 Z(4)=1000.
1580 LP(1)=3
1600 LP(2)=6
1620 LP(3)=4
1640 LP(4)=2
1680 LP(5)=5
1690 VLT$(1)="; "
1692 VLT$(2)="6;"
1694 VLT$(3)="2;"
1696 VLT$(4)="4;"
1698 VLT$(5)="1;"
1720 REM
1740 REM  --READ RANGE FROM UNIT 1
1760 REM
1762 PRINT "INPUT DATA FILENAME (XXXXXXX.DAT) PLEASE";
1764 INPUT F$
1766 OPEN F$ FOR INPUT AS FILE #1
1780 INPUT #1, X0,X1,Y0,Y1,Y02,Y12
1800 XL=X1-X0
1820 YL=Y1-Y0
1840 YL2=Y12-Y02
1880 REM
1900 REM --AXES LABELS
1920 USTR(1)=0.
1940 USTR(2)=.2
1960 USTR(3)=.4
1980 USTR(4)=.6
2000 USTR(5)=.8
2020 USTR(6)=1.
2040 REM --X-AXIS LABEL COORDINATES
2060 UX(1)=900.
2080 UX(2)=2200.
2100 UX(3)=3800.
2120 UX(4)=5400.
2140 UX(5)=7000.
2160 UX(6)=8650.
2180 UY=800.
```

```
4380 REM
4400 REM  ROUTINE TO CONVERT NORMALIZED COORDINATES INTO PLOTTER COORDINATES
4420 REM   SUBROUTINE PEER(X,Y,W,Z,PX,PY)
4440 REM   DIMENSION W(4),Z(4)
4460 PX=Z(1)*(X-W(1))/(W(2)-W(1))+Z(2)
4480 PY=Z(3)*(Y-W(3))/(W(4)-W(3))+Z(4)
4500 RETURN
4520 END
```

```
3300 IF (RRX = -1.234) THEN GOTO 3980
3320 REM  SELECTS PEN COLOR AND LINE-TYPE
3340 REM     1 : (RED)    _____
3360 REM     2 : (GREEN)  _____
3380 REM     3 : (VIOLET) _. _.
3400 REM     4 : (ORANGE) _. _.
3420 REM     5 : (BLUE)   . . . .
3440 PRINT " SI;SP";LP(L);";"
3460 PRINT " LT";VLT$(L)
3480 L=L+1
3500 I=0
3520 REM ------------------------------------------------
3540 REM  READ DATA CARDS FROM UNIT 1
3560 REM  EACH GROUP OF DATA CARDS ENDS WITH RY=-1.234
3580 INPUT #1, RX,RY
3600 RRX=RX
3620 IF (RY = -1.234) GOTO 3280
3640 REM  NORMALIZE, SUCH THAT [X0,X1] AND [Y0,Y1] BECOME [0,1]
3660 RX=(RX-X0)/XL
3680 RY=(RY-Y0)/YL
3700 X=RX
3720 Y=RY
3740 GOSUB 4460
3760 PX=PX
3780 PY=PY
3800 IF (I <> 0) GOTO 3860
3820 PRINT " PUPA ";PX;",";PY;";PD;"
3840 GOTO 3880
3860 PRINT " PA";PX;",";PY;";"
3880 I=I+1
3900 GOTO 3580
3920 REM ------------------------------------------------
3940 REM
3960 REM  WRITES NORMALIZATION INFORMATION
3980 TX=TTX(ITT)
4000 TY=TTY(ITT)
4020 PRINT " SI;SP5;"
4040 TY=TY-200.
4060 TY=TY-300.
4080 TY=TY-200.
4100 REM
4120 REM  READ END LABELS FROM UNIT 1
4140 K=1
4160 INPUT #1, AX,AY
4180 LINPUT #1, ASTR$
4200 PRINT " DI1,0;PUPA ";AX;",";AY
4220 PRINT " LB";ASTR$;" "
4240 GOTO 4160
4260 REM
4280 IF (ERR=11%) THEN RESUME 4300
4300 PRINT " PU;SP0;"
4320 CLOSE #1
4340 GOTO 4520
4360 REM
```

POLAR PLOT PROGRAM

```
100 REM POLAR.BAS
105 DIM W(4),TX(7),TY(7),LP(5)
110 DATA 1.,-1.,-1.,1.
120 READ W(1),W(2),W(3),W(4)
130 REM ON ERROR GOTO 1632
140 A1=1.745329E-2
150 A2=3.14159/180.
160 REM RADIAL RANGE OF PLOT (-40 TO +10 DBI)
170 A3=50.
180 REM RADIAL LABELS FOR THE POLAR PLOT
190 TR$(1)='   0'
200 TR$(2)=' -10'
210 TR$(3)=' -20'
220 TR$(4)=' -30'
230 TR$(5)=' -40'
240 TR$(6)=' +10'
250 TR$(7)=' DBI'
260 REM POSITIONS FOR RADIAL LABELS
270 DATA 3700.,3700.,3700.,3700.,3700.,3700.,3500.
280 READ TX(1),TX(2),TX(3),TX(4),TX(5),TX(6),TX(7)
290 DATA 1200.,1850.,2620.,3380.,4140.,450.,450.
300 READ TY(1),TY(2),TY(3),TY(4),TY(5),TY(6),TY(7)
310 DATA "; ","2;","6;","4;","1;"
320 READ VLT$(1),VLT$(2),VLT$(3),VLT$(4),VLT$(5)
330 DATA 3,4,6,2,5
340 READ LP(1),LP(2),LP(3),LP(4),LP(5)
350 PRINT "INPUT YOUR INPUT FILE NAME (XXXXXX.DAT) PLEASE";
360 INPUT F$
370 OPEN F$ FOR INPUT AS FILE #1
380 REM
390 REM
400 REM INITIALIZE : SI  NECESSARY FOR CORRECT LETTER SIZE
410 REM      SP5 FOR (BLUE) PEN
420 REM      DT  TO SET " " AS THE END-STRING CHARACTER
430 PRINT "IN;SI;SP5;DT "
440 REM READ LABELS FROM (FILE) UNIT 1, WHERE
450 REM   (RX,RY) ARE THE COORDINATES TO PRINT LABELS
460 REM   ASTR IS LABEL
470 INPUT #1, RX,RY,ASTR$
480 REM DI0,-1; IS THE DIRECTION OF LABEL PRINTING
490 PRINT " DI0,-1;PUPA";RX;",";RY;",PD;"
500 REM " " USED AS TERMINATOR FOR STRING, AS DEFINED BY "DT"
510 PRINT " LB";ASTR$;" "
520 GOTO 470
530 REM PRINT RADIAL LABELS
540 FOR L=1 TO 7
550 PRINT " DI0,-1;PUPA";TX(L);",";TY(L);",PD;"
560 PRINT " LB";TR$(L);" "
570 NEXT L
580 REM
590 REM DRAW THE POLAR PLOT BACKGROUND (USE BLACK PEN)
600 PRINT " SP1;"
```

```
610 REM SET BOUNDARIES
620 PRINT " IP2650,1825,7650,6825;"
630 R=0.
640 FOR K=1 TO 5
650 REM 5 CIRCLES OF RADII .2, .4, .6, .8, 1.
660 R=R+.2
670 FOR J=0 TO 360 STEP 5
680 REM ROTATE ANGLE BY 90 DEGREES
690 X=R*COS((J-90)*A1)
700 Y=R*SIN((J-90)*A1)
710 REM CONVERT COORDINATES TO PLOTTER COORDINATES
720 GOSUB 1560
730 REM CALL PEER(X,Y,W,PX,PY)
740 IF (J <> 0) THEN GOTO 770
750 PRINT " PUPA";PX;",";PY;";PD;"
760 GOTO 780
770 PRINT " PA";PX;",";PY;";"
780 NEXT J
790 NEXT K
800 REM
810 REM DRAW CROSSED LINES
820 X1=0.
830 Y1=-1.
840 X2=0.
850 Y2=1.
860 REM CALL PEER(X1,Y1,W,PX1,PY1)
870 X=X1
880 Y=Y1
890 GOSUB 1560
900 PX1=PX
910 PY1=PY
920 REM CALL PEER(X2,Y2,W,PX2,PY2)
930 X=X2
940 Y=Y2
950 GOSUB 1560
960 PX2=PX
970 PY2=PY
980 PRINT " PUPA";PX1;";";PY1;";";"PDPA";PX1;";";PY2;";"
990 REM 'PDPA',F7.1,',',F7.1,';')
1000 REM CALL PEER(Y1,X1,W,PX1,PY1)
1010 X=Y1
1020 Y=X1
1030 GOSUB 1560
1040 PX1=PX
1050 PY1=PY
1060 REM CALL PEER(Y2,X2,W,PX2,PY2)
1070 X=Y2
1080 Y=X2
1090 GOSUB 1560
1100 PX2=PX
1110 PY2=PY
1120 PRINT " PUPA";PX1;";";PY1;";";"PDPA";PX1;";";PY2;";"
1130 REM
1140 REM SETS PEN COLOR AND LINETYPE (LIMIT OF 5)
```

```
1150 L=1
1160 REM PEN COLOR
1170 IF (L > 5) THEN GOTO 1460
1180 IF (ANG = -1.234) THEN GOTO 1460
1190 REM ALLOW 5 PLOTS AND/OR STOP WHEN ANGLE OF -1.234 ENCOUNTERED
1200 PRINT " SI;SP";LP(L);";"
1210 REM BOUNDARIES
1220 PRINT " IP2650,1825,7650,6825;"
1230 REM LINETYPE
1240 PRINT " LT";VLT$(L)
1250 L=L+1
1260 REM READ FROM UNIT 2 ANGLE (DEGREES) AND POWER GAIN (PG+40)
1270 I=0
1280 INPUT #1, ANG,RAD
1290 REM ASSUME EACH GROUP OF DATA ENDS WITH CARD OF RADIUS -1.234
1300 IF (RAD = -1.234) THEN GOTO 1170
1310 REM ROTATE ANGLE BY 90 DEGREES AND CONVERT TO PLOTTER COORDINATES
1320 ANG=ANG-90.
1330 X=(RAD/A3)*COS(ANG*A2)
1340 Y=(RAD/A3)*SIN(ANG*A2)
1350 GOSUB 1560
1360 REM CALL PEER(X,Y,W,PX,PY)
1370 IF (I <> 0) THEN GOTO 1400
1380 PRINT " PUPA";PX;",";PY;";PD;"
1390 GOTO 1410
1400 PRINT " PA";PX;",";PY;";"
1410 I=I+1
1420 ANG=ANG-90.
1430 GOTO 1280
1440 REM
1450 REM WRITES ANY END LABELS FROM UNIT 2
1460 PRINT "SI;SP5;"
1470 REM SET PEN COLOR (SI NEEDED FOR CORRECT LETTER SIZE)
1480 INPUT #1, RX,RY,ASTR$
1490 PRINT " DI0,-1;PUPA";RX;",";RY;",PD;"
1500 PRINT " LB";ASTR$;" "
1510 GOTO 1480
1520 REM IF (ERR=11%) THEN RESUME 1634
1530 PRINT " PU;SP0;"
1540 STOP
1550 REM
1560 REM ROUTINE TO CONVERT NORMALIZED COORDINATES TO PLOTTER COORDINATES
1570 REM SUBROUTINE PEER(X,Y,W,PX,PY)
1580 PX=7650.*(X-W(3))/(W(4)-W(3))
1590 PY=7650.*(Y-W(2))/(W(1)-W(2))
1600 RETURN
1610 END
```

SMITH PLOT PROGRAM

```
10 DIM VSWR(4)
20 PRINT "ENTER VSWR 2-5";
30 INPUT V
40 V=V-1
50 CLS
60 DATA 0.0,224.0,0.0,512.0,1.0,-1.0,-1.0,1.0
70 DATA 1.745329E-2,0.33871,0.5,0.6,0.6693548
80 READ VYT,VYB,VXL,VXR,WYT,WYB,WXL,WXR
90 READ RADIAN,VSWR(1),VSWR(2),VSWR(3),VSWR(4)
100 REM DRAW RESISTANCE AXIS
110 FOR K=0 TO 3 STEP 3
120 FOR I=-30 TO -2
130 GOSUB 500
140 IF (K=0) AND (I=-30) THEN LINE (XS,YS)-(XS,YS) ELSE LINE -(XS,YS)
150 NEXT I
160 I=-2!
170 FOR M=-50 TO 50
180 GOSUB 500
190 LINE -(XS,YS)
200 I=I+.04
210 NEXT M
220 FOR I=2 TO 30
230 GOSUB 500
240 LINE -(XS,YS)
250 NEXT I
260 NEXT K
270 REM DRAW REACTANCE AXIS
280 FOR I=-1 TO 1
290 K=0
300 FOR J=0 TO 20
310 GOSUB 500
320 IF (J=0) THEN LINE (XS,YS)-(XS,YS) ELSE LINE -(XS,YS)
330 K=K+.1
340 NEXT J
350 FOR J=2 TO 30
360 K=J
370 GOSUB 500
380 LINE -(XS,YS)
390 NEXT J
400 NEXT I
410 REM DRAW VSWR DEFINITION CIRCLE (2-5)
420 FOR J=0 TO 360 STEP 2
430 XW=VSWR(V)*COS(J*RADIAN)
440 YW=VSWR(V)*SIN(J*RADIAN)
450 GOSUB 580
460 IF (J=0) THEN LINE (XS,YS)-(XS,YS) ELSE LINE -(XS,YS)
470 NEXT J
480 INPUT A$
490 STOP
500 REM SUBROUTINE
510 R=K
520 A=R-1!
```

```
530 B=I
540 C=R+1!
550 D=B
560 XW=(A*C+B*D)/(C*C+D*D)
570 YW=(B*C-A*D)/(C*C+D*D)
580 XS=((VXR-VXL)/(WXR-WXL))*(XW-WXL)+VXL
590 YS=((VYT-VYB)/(WYT-WYB))*(YW-WYB)+VYB
600 RETURN
610 END
```

APPENDIX C
SAMPLE DATA FILES

The following are listings of data files for each of the available graph types. Typically, a data file consists of three parts: (1) a line containing possible X/Y range values; (2) data sets (possibly multiple) or lines of data; and (3) label information.

The sample files below were used to create the plots in figures 1 through 7. You should use the example data files as guides in creating your own GRAPS data files.

LINEAR DATA

```
0,1500,0,50
40,  1.140
80,  2.165
160, 3.907
320, 7.285
640, 13.545
1280,     17.702
1.234,    -1.234
40.,  1.353
80.,  2.513
160.,     4.704
320.,     8.962
640.,     17.903
1280.,    28.797
1.234,    -1.234
40.,  1.429
80.,  2.676
160.,     5.051
320.,     9.718
640.,     19.928
1280.,    35.433
1.234,    -1.234
40.,  1.442
80.,  2.713
160.,     5.147
320.,     9.965
640.,     20.665
1280.,    38.180
-1.234, -1.234
3,14
EFFICIENCY  VS. RADIAL LENGTH (FT)
4,14
(EFFICIENCY BASED ON AVERAGE GAIN)
5,13
1 KFT TOP LOADED MONOPOLE AT  25 KHZ
6,16
12 TOP HAT RADIALS;  H'/H = .6
7,17
BURIED RADIAL GROUND SYSTEM
8,21
RADIALS BURIED 1 FT.
9,14
EPSILON = 10., SIGMA = .001 MHOS/M
18,50
12 RADIALS _____
19,50
24 RADIALS _____
20,50
48 RADIALS ____. ____
21,50
96 RADIALS _____
25,35
```

RADIAL LENGTH (FT)
1,3
EFFICIENCY (%)

BILINEAR DATA

0.,600.,0.,2.,0.,15.
1.60191, 1.659
5.04602, 1.654
9.00674, 1.647
13.5616, 1.641
18.7996, 1.633
24.8234, 1.626
31.7507, 1.617
39.7171, 1.608
48.8785, 1.597
59.4141, 1.586
71.53, 1.573
85.4634, 1.560
101.487, 1.546
119.913, 1.530
141.104, 1.515
163.774, 1.500
184.964, 1.488
203.391, 1.479
219.415, 1.473
233.348, 1.468
245.464, 1.464
255.999, 1.461
265.161, 1.459
273.127, 1.457
280.055, 1.455
286.078, 1.454
291.316, 1.454
295.871, 1.453
299.832, 1.453
303.276, 1.452
306.475, .1209
309.908, .1206
313.855, .1201
318.396, .1193
323.617, .1184
329.621, .1172
336.526, .1156
344.466, .1137
353.598, .1113
364.099, .1082
376.176, .1044
390.064, .09963
406.035, .09360
424.402, .08598
443.403, .07732
460.561, .06879
475.481, .06081
488.455, .05344
499.736, .04668
509.546, .04052
518.076, .03495

525.494, .02933
531.944, .02542
537.553, .02139
542.431, .01777
546.672, .01455
550.36, .01167
553.567, .009091
556.355, .006783
558.78, .004708
560.889, .002815
562.722, .001013
1.234, -1.234
1.60191, 6.114
5.04602, 11.25
9.00674, 9.723
13.5616, 9.003
18.7996, 8.490
24.8234, 8.073
31.7507, 7.706
39.7171, 7.363
48.8785, 7.028
59.4141, 6.686
71.53, 6.328
85.4634, 5.941
101.487, 5.512
119.913, 5.029
141.104, 4.486
163.774, 3.859
184.964, 3.268
203.391, 2.806
219.415, 2.434
233.348, 2.130
245.464, 1.878
255.999, 1.665
265.161, 1.481
273.127, 1.320
280.055, 1.173
286.078, 1.036
291.316, .9023
295.871, .7658
299.832, .6156
303.276, .4019
306.475, .4570
309.908, .7567
313.855, .9328
318.396, 1.081
323.617, 1.216
329.621, 1.348
336.526, 1.481
344.466, 1.619
353.598, 1.765
364.099, 1.922
376.176, 2.096
390.064, 2.291

406.035, 2.513
424.402, 2.770
443.403, 3.040
460.561, 3.291
475.481, 3.518
488.455, 3.723
499.736, 3.909
509.546, 4.080
518.076, 4.237
525.494, 4.383
531.944, 4.520
537.553, 4.652
542.431, 4.781
546.672, 4.910
550.36, 5.044
553.567, 5.190
556.355, 5.359
558.78, 5.573
560.889, 5.898
562.722, 6.698
-1.234, -1.234
3,25
CURRENT AND CHARGE DISTRIBUTION
4,23
1 KFT TOP LOADED MONOPOLE AT 25 KHZ
5,22
12 TOP HAT WIRES (H'/H=.6) AT 45 DEG
6,33
NO CORONA RINGS
13,48
1 VOLT SOURCE
14,48
PERFECT GROUND
13,20
CURRENT _____
14,20
CHARGE _____
25,35
DISTANCE (M)
1,1
CURRENT (MA)
1,50
CHARGE (COULOMBS/M x 10e-12)

LOGLINEARV DATA

```
10000,      1E+08
500,   1500
2.14E+07,  500
1.02E+07,  600
5420000,   700
3120000,   800
1910000,   900
1230000,   1000
816000,    1100
561000,    1200
395000,    1300
284000,    1400
208000,    1500
1.234,      -1.234
1.08E+07,  500
5140000,   600
2710000,   700
1550000,   800
942000,    900
599000,    1000
395000,    1100
268000,    1200
186000,    1300
132000,    1400
94500,      1500
1.234,      -1.234
7510000,   500
3540000,   600
1860000,   700
1060000,   800
635000,    900
400000,    1000
260000,    1100
174000,    1200
119000,    1300
82900,      1400
58400,      1500
1.234,      -1.234
554789.7,  1000
-1.234,    -1.234
3,15
X^2/R VS ANTENNA HEIGHT AT 25 KHZ
4,27
H/H' = .6
5,19
PERFECTLY CONDUCTING EARTH
6,24
TOP HAT RADIALS
18,12
6 _____
19,12
12 _____
```

20,12
24 ____. ____
21,12
1,2
X^2/R (OHMS)
25,22
ANTENNA HEIGHT (FT)

LOGLINEARH DATA

```
10000,      1E+08
500,   1500
2.14E+07,  500
1.02E+07,  600
5420000,   700
3120000,   800
1910000,   900
1230000,   1000
816000,    1100
561000,    1200
395000,    1300
284000,    1400
208000,    1500
1.234,     -1.234
1.08E+07,  500
5140000,   600
2710000,   700
1550000,   800
942000,    900
599000,    1000
395000,    1100
268000,    1200
186000,    1300
132000,    1400
94500,      1500
1.234,     -1.234
7510000,   500
3540000,   600
1860000,   700
1060000,   800
635000,    900
400000,    1000
260000,    1100
174000,    1200
119000,    1300
82900,      1400
58400,      1500
1.234,     -1.234
554789.7,  1000
-1.234,  -1.234
3,35
X^2/R VS ANTENNA HEIGHT AT 25 KHZ
4,47
H/H' = .6
5,38
PERFECTLY CONDUCTING EARTH
6,44
TOP HAT RADIALS
18,12
6 _____
19,12
12 _____
```

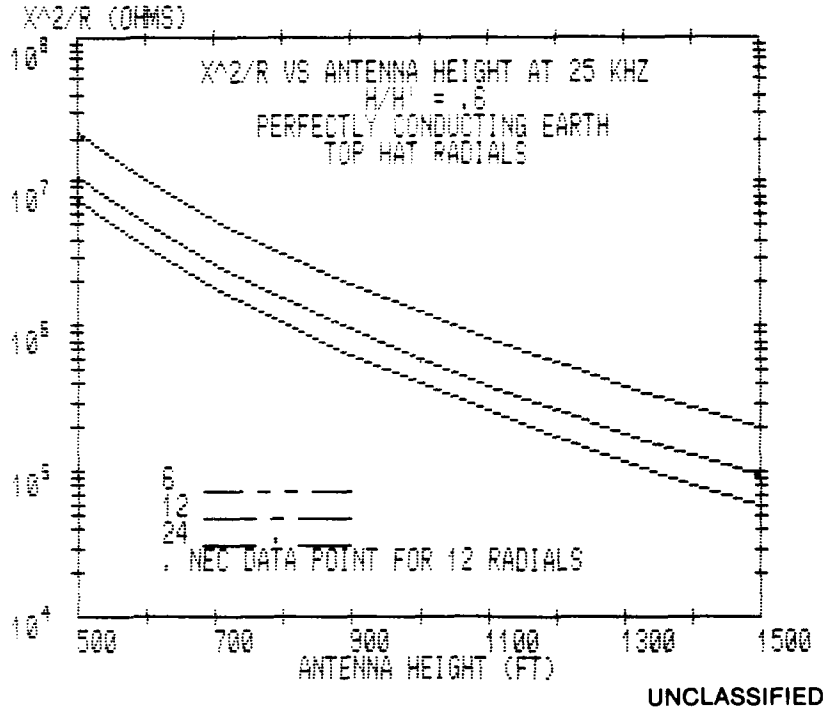


Figure 10

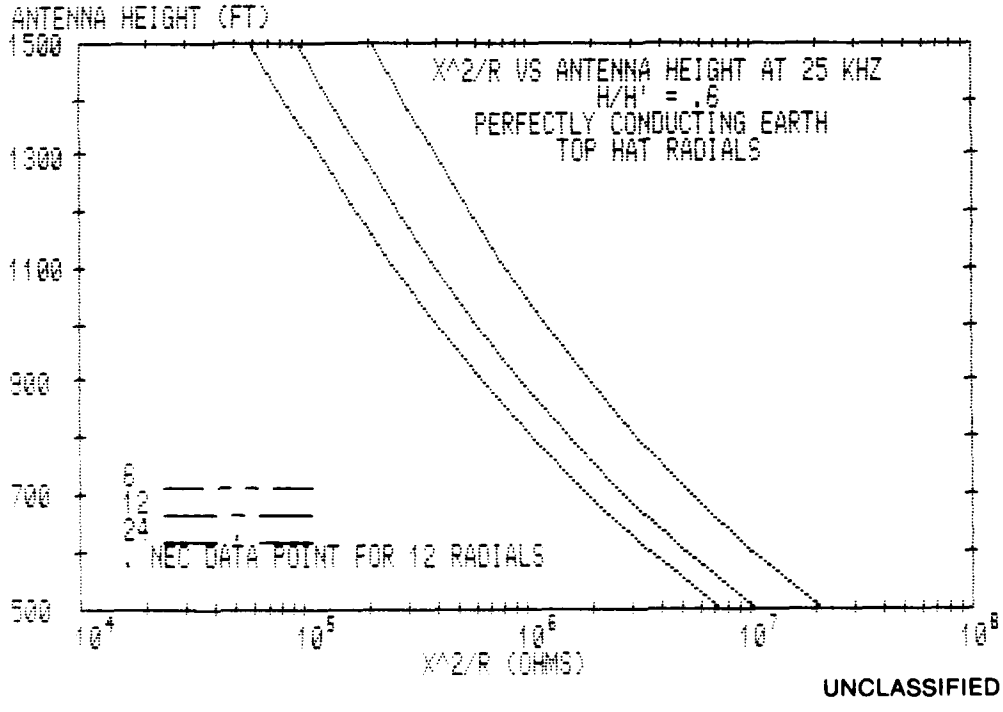


Figure 11

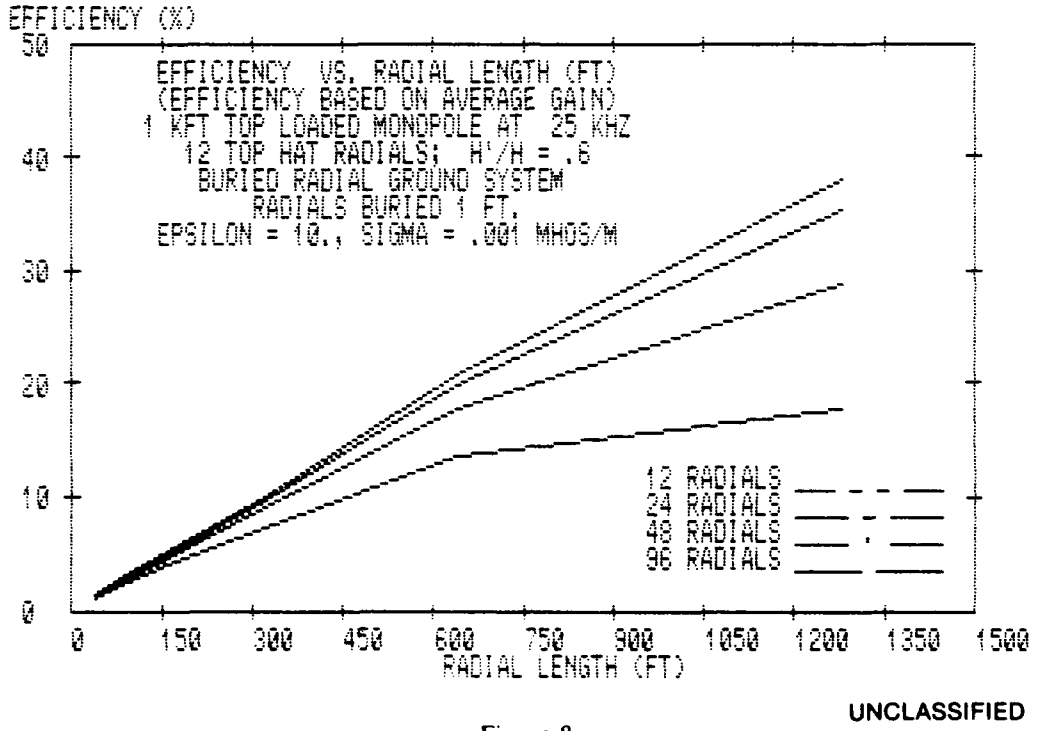


Figure 8

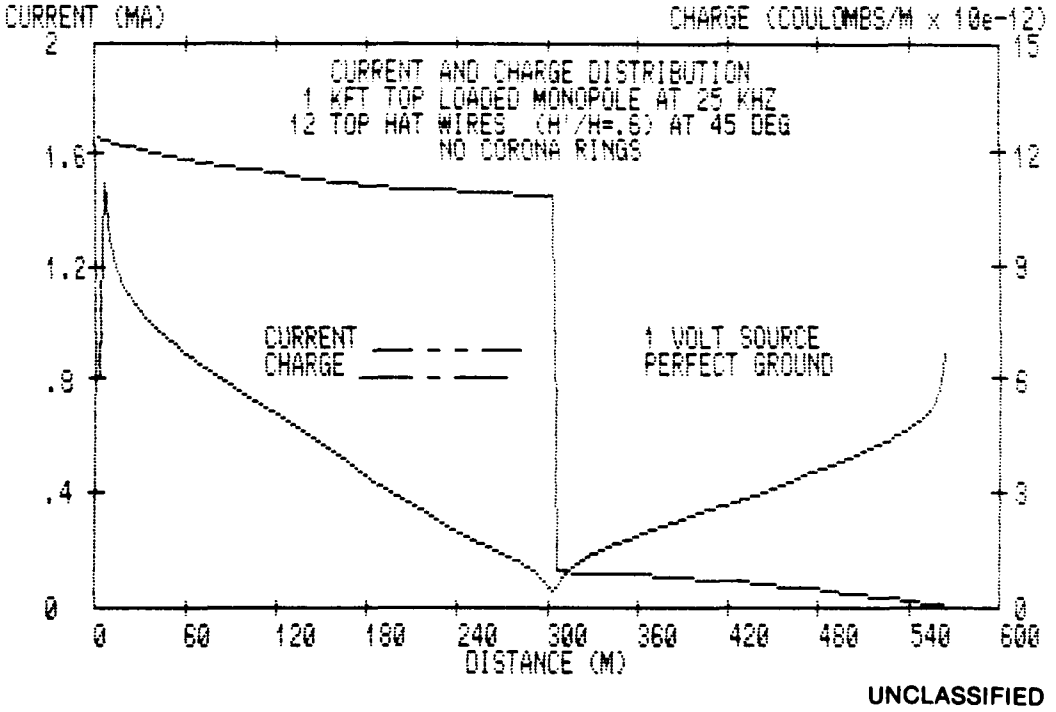


Figure 9

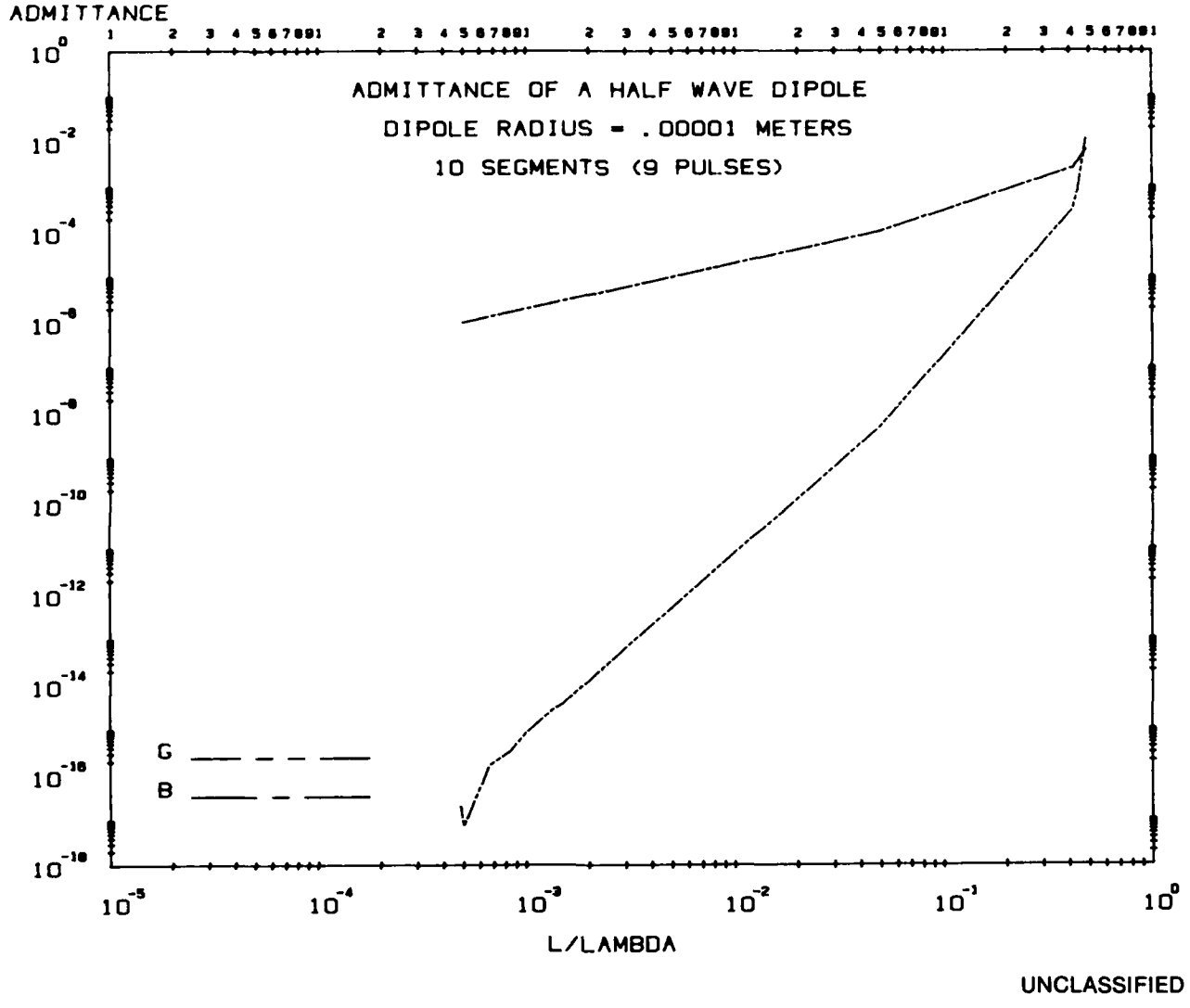
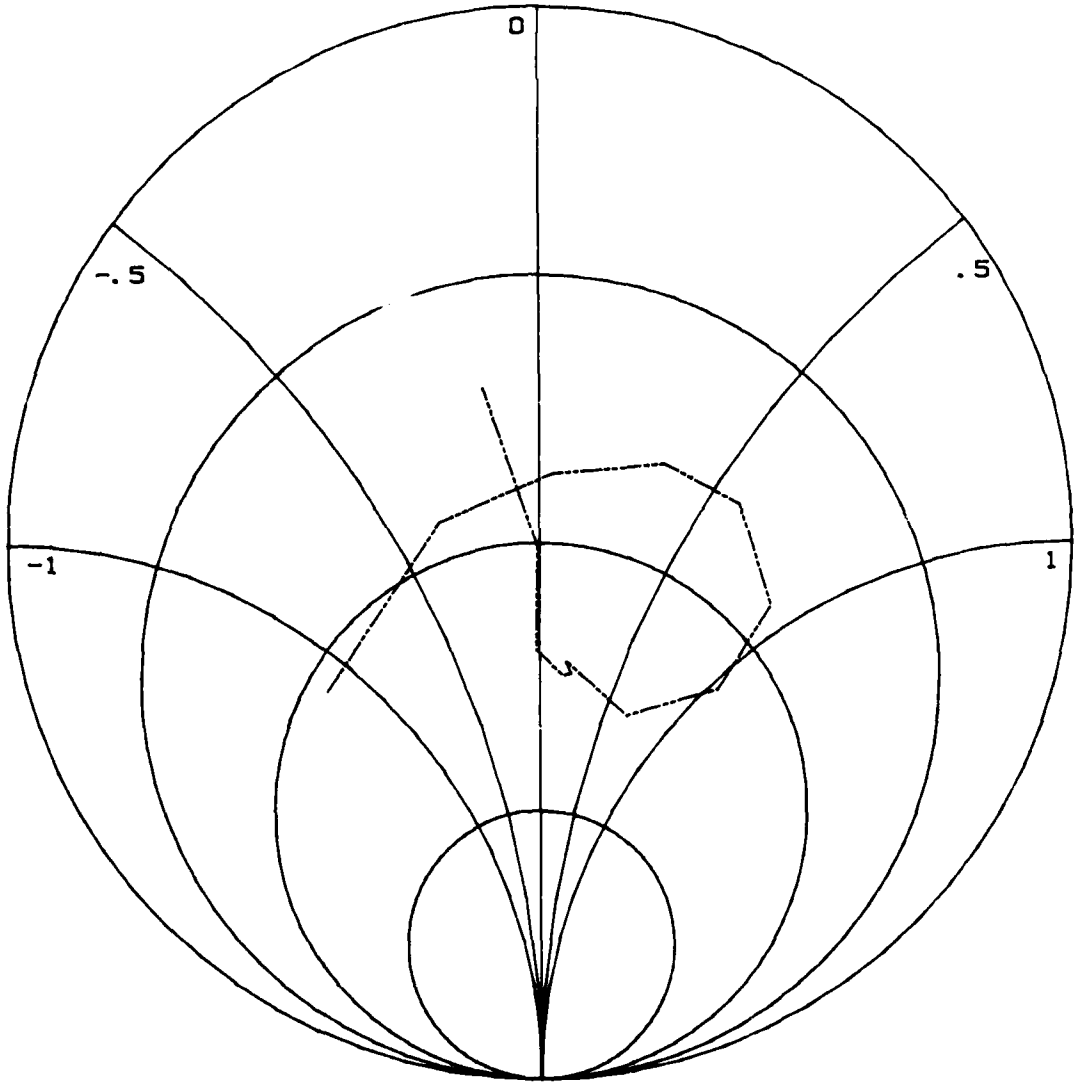

ADMITTANCE
ADMITTANCE OF A HALF WAVE DIPOLE
DIPOLE RADIUS = .00001 METERS
10 SEGMENTS (9 PULSES)
G
B
L/LAMBDA
UNCLASSIFIED

Figure 7

BROADBAND ANTENNA

3: 1 VSWR DEFINITION CIRCLE

FREQUENCY IN MHZ

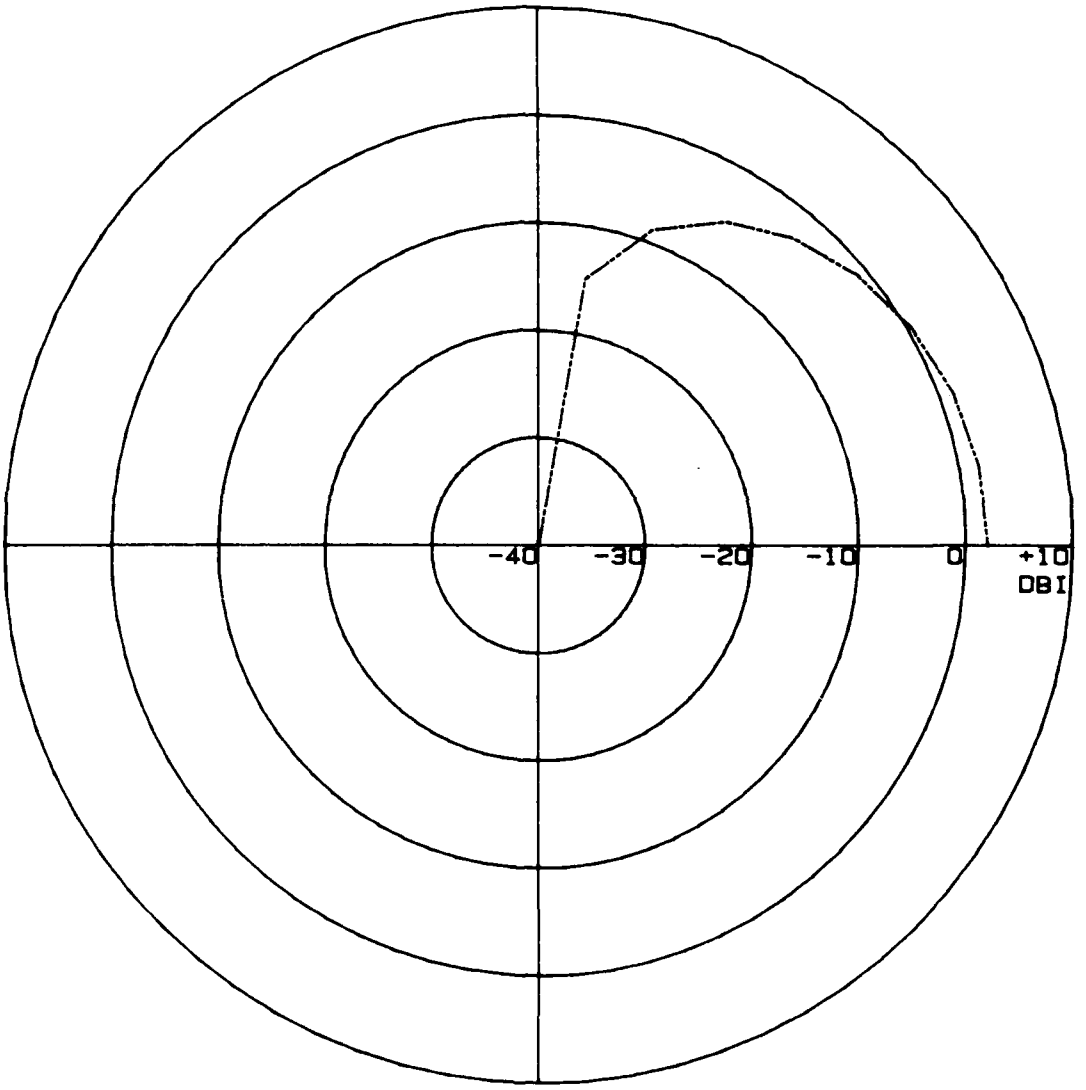


UNCLASSIFIED

Figure 6

10.67 METER MONOPOLE AT 10 MHZ

WIRE RADIUS = .0508 METERS



UNCLASSIFIED

Figure 5

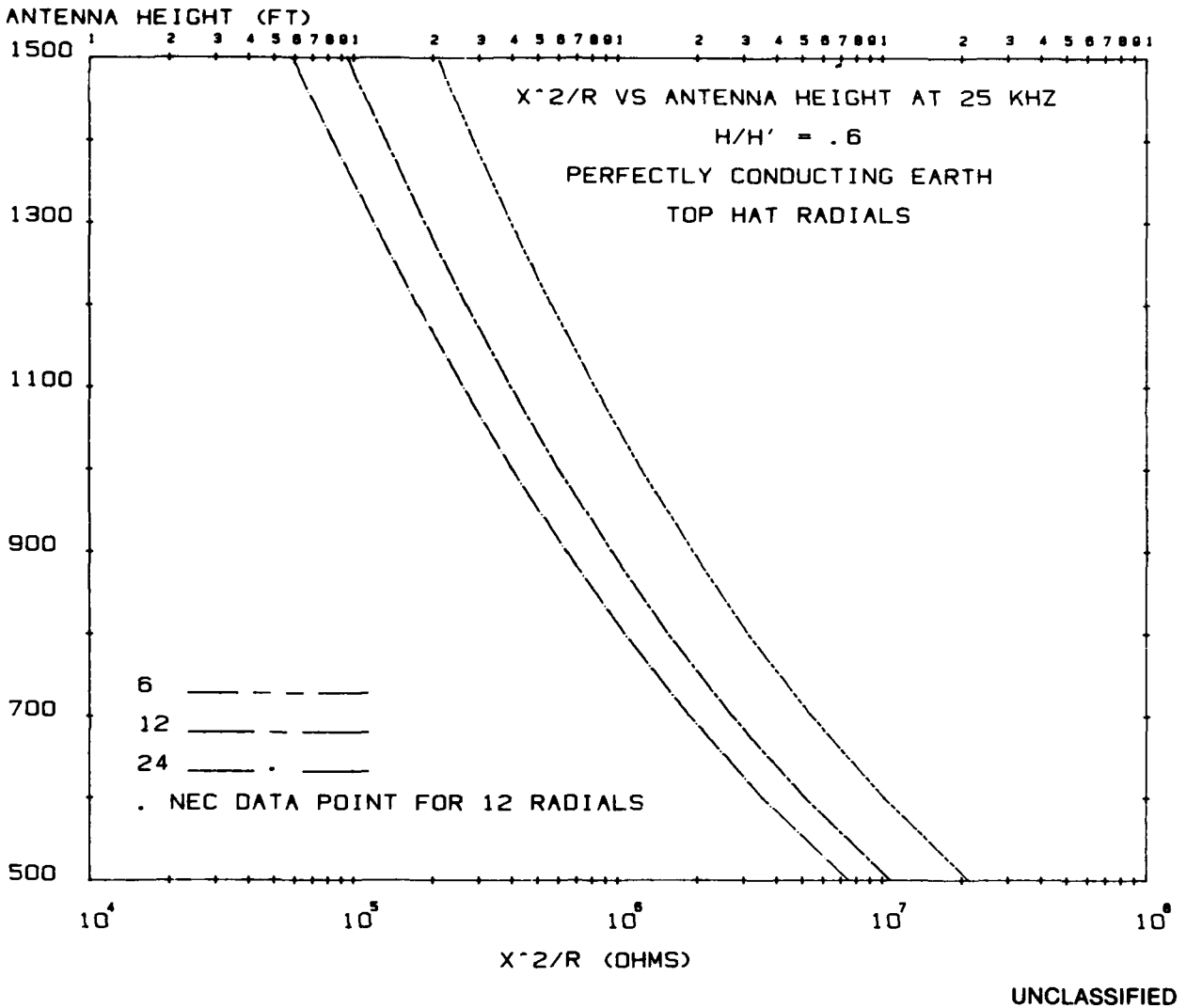

ANTENNA HEIGHT (FT)
1500
1300
1100
900
700
500
X^2/R VS ANTENNA HEIGHT AT 25 KHZ
H/H' = .6
PERFECTLY CONDUCTING EARTH
TOP HAT RADIALS
6
12
24
. NEC DATA POINT FOR 12 RADIALS
10^4
10^5
10^6
10^7
10^8
X^2/R (OHMS)
UNCLASSIFIED

Figure 4

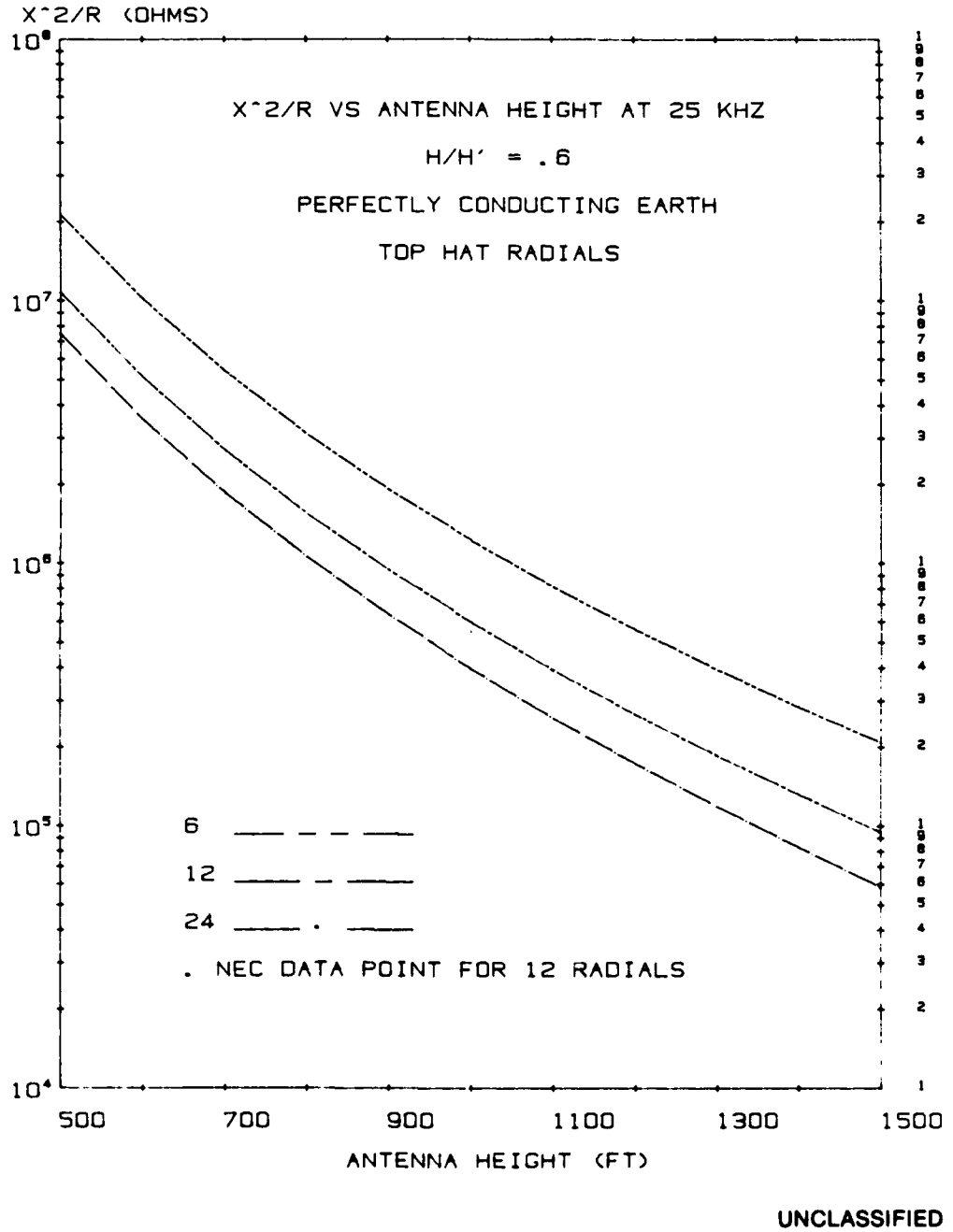

X^2/R (OHMS)
X^2/R VS ANTENNA HEIGHT AT 25 KHZ
H/H' = .6
PERFECTLY CONDUCTING EARTH
TOP HAT RADIALS
6
12
24
. NEC DATA POINT FOR 12 RADIALS
ANTENNA HEIGHT (FT)
UNCLASSIFIED

Figure 3

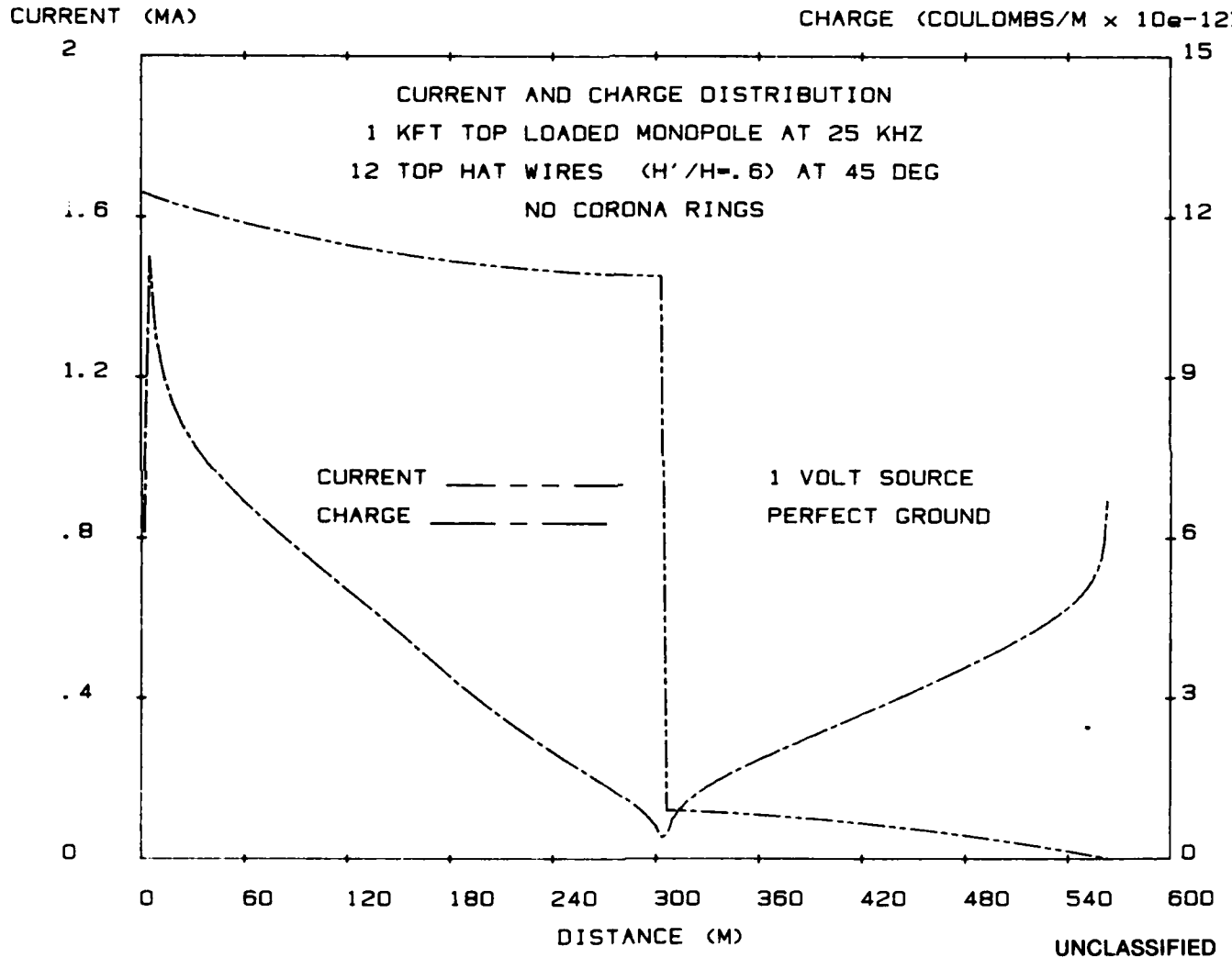

CURRENT (MA)
CHARGE (COULOMBS/M x 10e-12)
2
1.6
1.2
.8
.4
0
15
12
9
6
3
0
CURRENT AND CHARGE DISTRIBUTION
1 KFT TOP LOADED MONOPOLE AT 25 KHZ
12 TOP HAT WIRES (H'/H=.6) AT 45 DEG
NO CORONA RINGS
CURRENT
CHARGE
1 VOLT SOURCE
PERFECT GROUND
0
60
120
180
240
300
360
420
480
540
600
DISTANCE (M)
UNCLASSIFIED

Figure 2

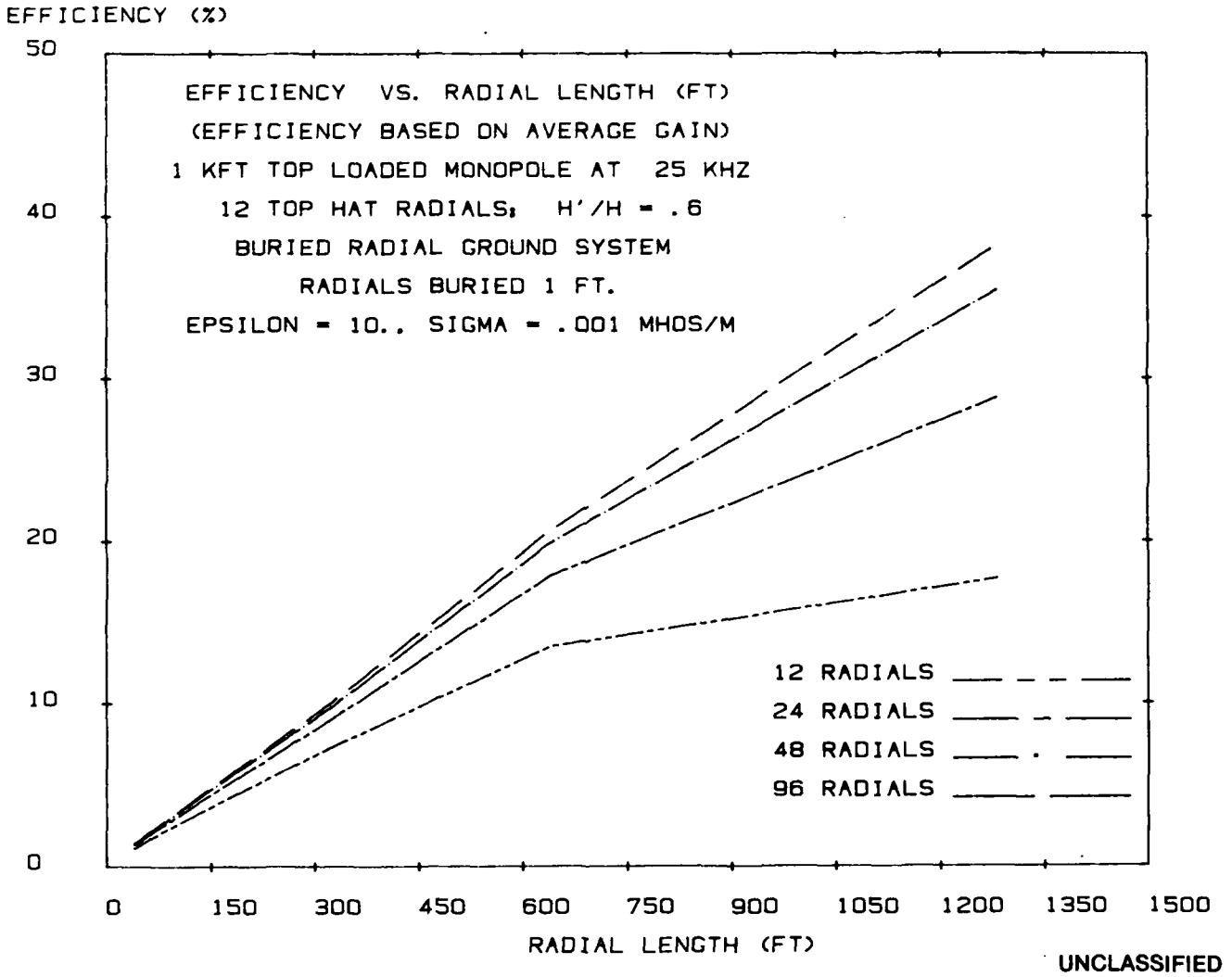

EFFICIENCY (%)
50
40
30
20
10
0
EFFICIENCY VS. RADIAL LENGTH (FT)
(EFFICIENCY BASED ON AVERAGE GAIN)
1 KFT TOP LOADED MONOPOLE AT 25 KHZ
12 TOP HAT RADIALS; H'/H = .6
BURIED RADIAL GROUND SYSTEM
RADIALS BURIED 1 FT.
EPSILON = 10., SIGMA = .001 MHOS/M
12 RADIALS
24 RADIALS
48 RADIALS
96 RADIALS
0
150
300
450
600
750
900
1050
1200
1350
1500
RADIAL LENGTH (FT)
UNCLASSIFIED

Figure 1

APPENDIX D
PRINTER INSTALLATION

The GRAPS system recognizes a variety of graphics printers. Since GRAPS uses the *print screen* interrupt to print the image displayed upon the PC's screen, special software is required to process the interrupt and send the screen information to the graphics printer. In addition, since different printers use different commands to print graphics information (graphical images), different interrupt routines are required. These routines are particular to each type of printer. That is, there is an interrupt routine for the *Epson* line of printers, there is an interrupt routine for the *IDS* printer line, etc.

The design of GRAPS requires that the printer interrupt routine be resident before GRAPS is loaded.[1] This means that there must be some preprocessing that performs this operation. There are special programs that do just this; that is, programs that load the print screen software and redirect interrupts to the new location. All that is required is that the program be executed before the user enters GRAPS. The distribution software includes a batch file that first runs this special program, and then executes GRAPS.[2]

GRAPS is "delivered" with the default printer type set to **IDS**. This means that any *IDS*-compatible printer will work with GRAPS without any modifications to the software. **To use other printer types, the user must make sure that, before entering GRAPS, the appropriate printer interrupt routine program is executed.** Thus, if I wanted to use an *Epson* printer, I would change the steps outlined in the *Operation* section to include the following:

```
cd graps
pscepson
basica graps .
```

The second line, **pscepson**, runs a program that installs the printer interrupt routine for *Epson* printers. Now, *print screen* interrupts will use this routine to process the screen image. (The *Print Graph* function of GRAPS uses the *print screen* interrupt.)

Any printer type for which there is a corresponding printer interrupt routine program can be used with GRAPS. You simply must know which program goes with which type of printer, and execute this program prior to entering GRAPS.

---

[1] By "resident", I mean that the software be loaded into high memory, and that the *print screen* interrupt vector be set to the address of the new interrupt routine.

[2] The name of the batch file is **rg.bat**. The name of the printer interrupt routine program is **pscids.com**. All printer interrupt programs begin with the three letters *psc* and are followed by an abbreviation for the type of printer, e.g., *ids* for *IDS* printers.

LOGLOG DATA

1.0E-5,1.0,1.0E-18,1.0
8.3389E-4, .315635E-15
6.6711E-4, .158849E-15
1.234, -1.234
8.3389E-4, .166269E-5
6.6711E-4, .133015E-5
-1.234, -1.234
1,3
ADMITTANCE
25,36
L/LAMBDA
3,24
ADMITTANCE OF A HALF WAVE DIPOLE
4,26
DIPOLE RADIUS = .00001 METERS
5,29
10 SEGMENTS (9 PULSES)
20,12
G _____
21,12
B _____

SMITH DATA

1.118, -1.171
1.279, 1.043
1.794, .67
1.552, .151
1.63, .197
1.652, .147
1.498, -.023
1.02, -.01
-1.234, -1.234
1,22
BROADBAND ANTENNA
2,17
3:1 VSWR DEFINITION CIRCLE
3,22
FREQUENCY IN MHZ

POLAR DATA

```
0,    -40.
10,   -14.80869
20,   -8.768149
30,   -5.238006
40,   -2.775642
50,   -.9580317
60,   .3795721
70,   1.307637
80,   1.856538
90,   2.038439
-1.234, -1.234
1,15
10.67 METER MONOPOLE AT 10 MHZ
2,17
WIRE RADIUS = .0508 METERS
```

20,12
24 ____. ____
21,12
1,4
ANTENNA HEIGHT (FT)
25,34
X^2/R (OHMS)

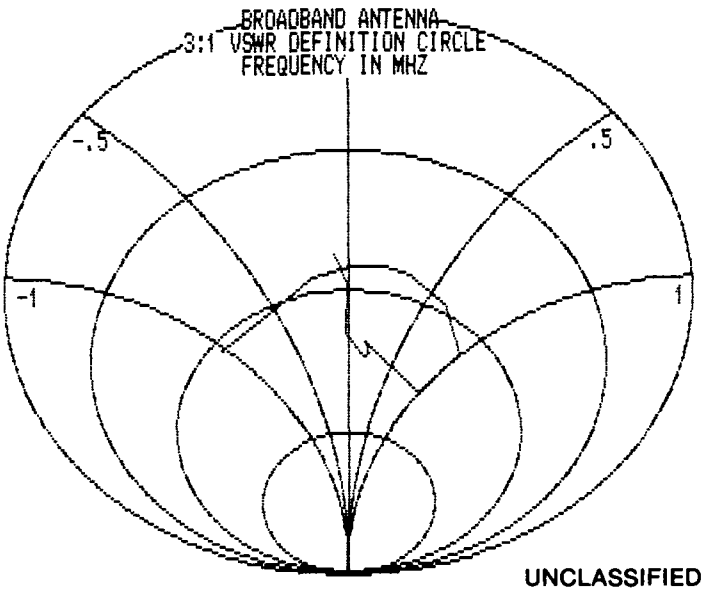


Figure 12

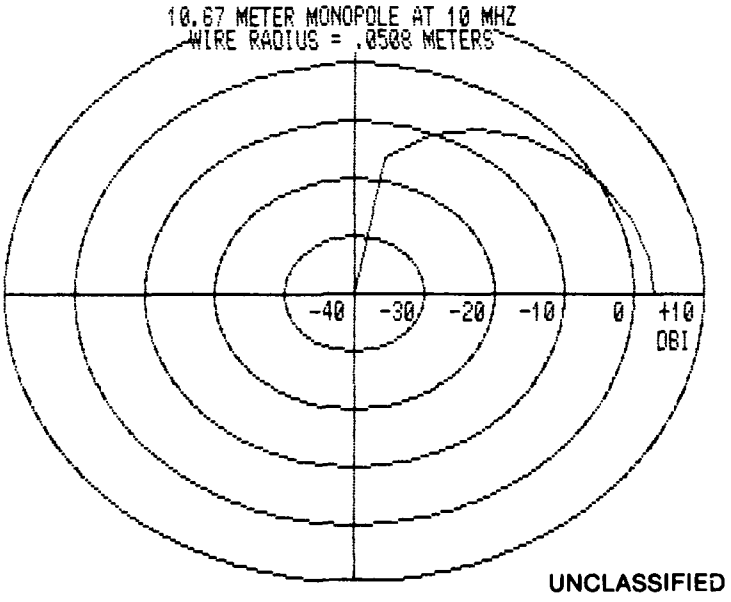


Figure 13

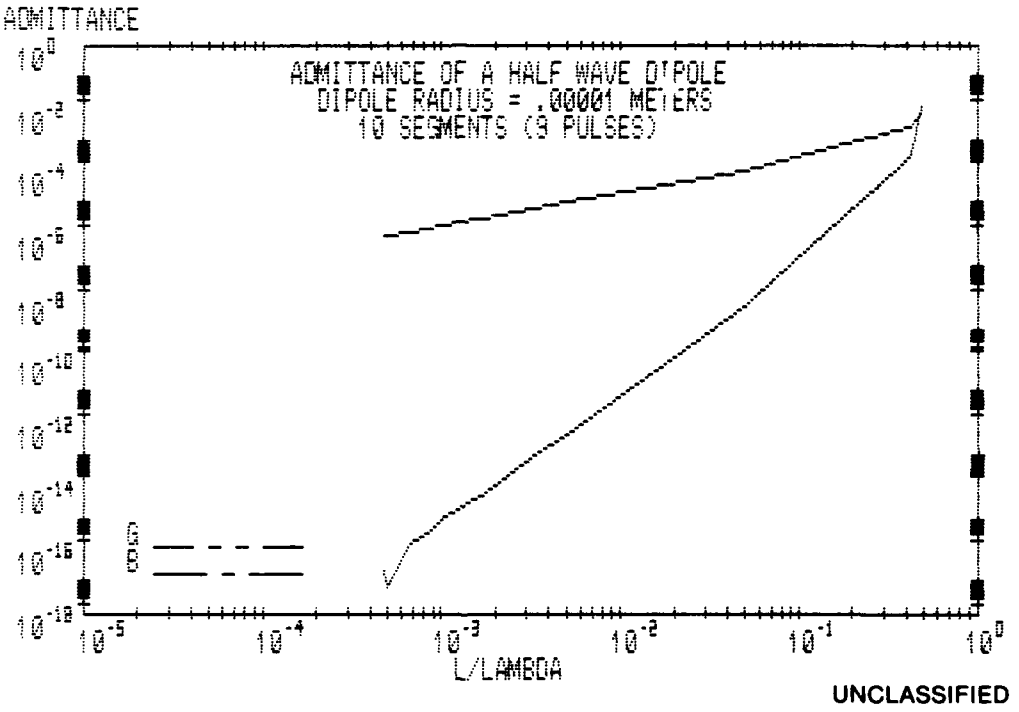

ADMITTANCE
ADMITTANCE OF A HALF WAVE DIPOLE
DIPOLE RADIUS = .00001 METERS
10 SEGMENTS (9 PULSES)
G
B
L/LAMBDA
UNCLASSIFIED

Figure 14

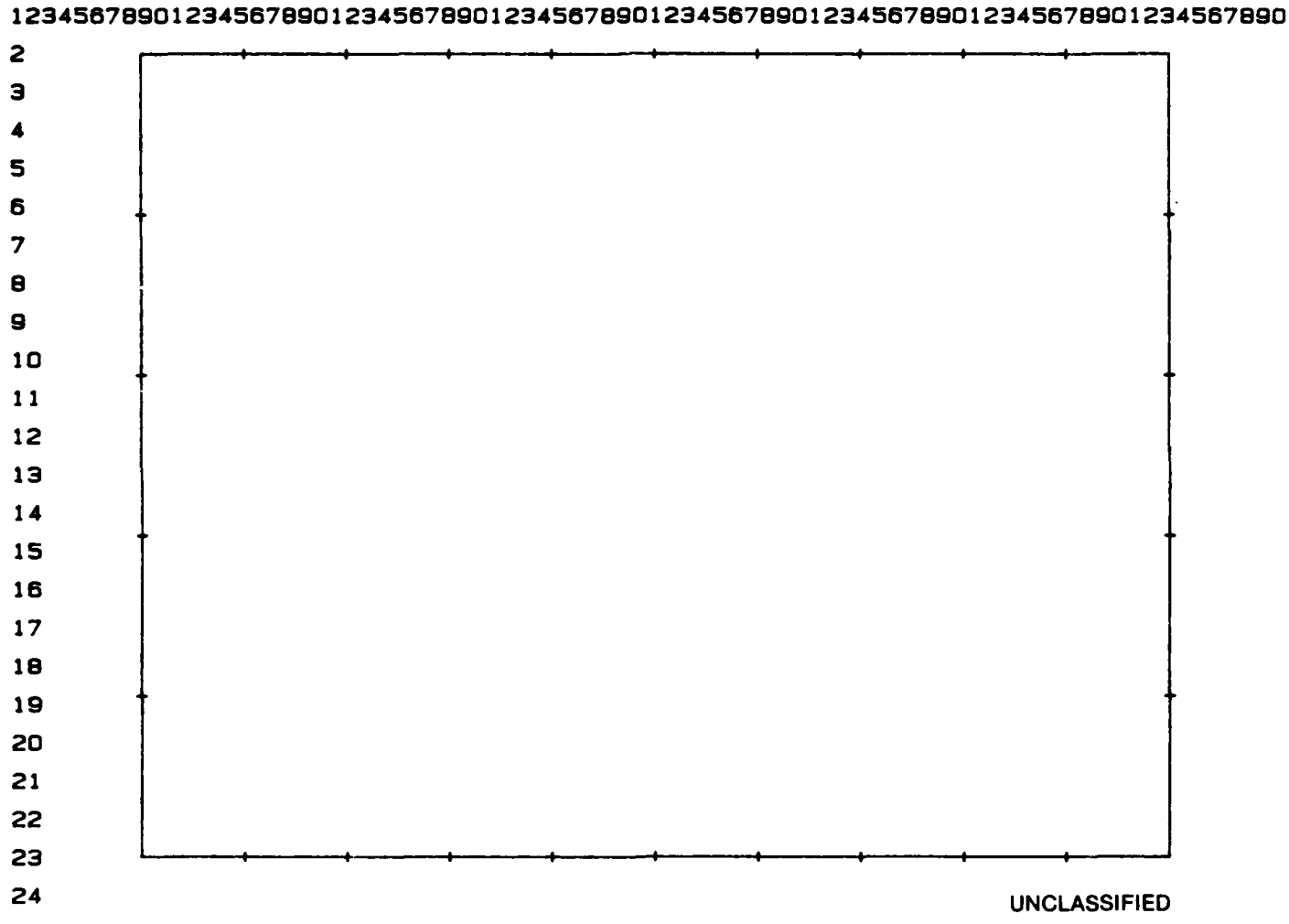

1234567890123456789012345678901234567890123456789012345678901234567890
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
UNCLASSIFIED

Figure 15

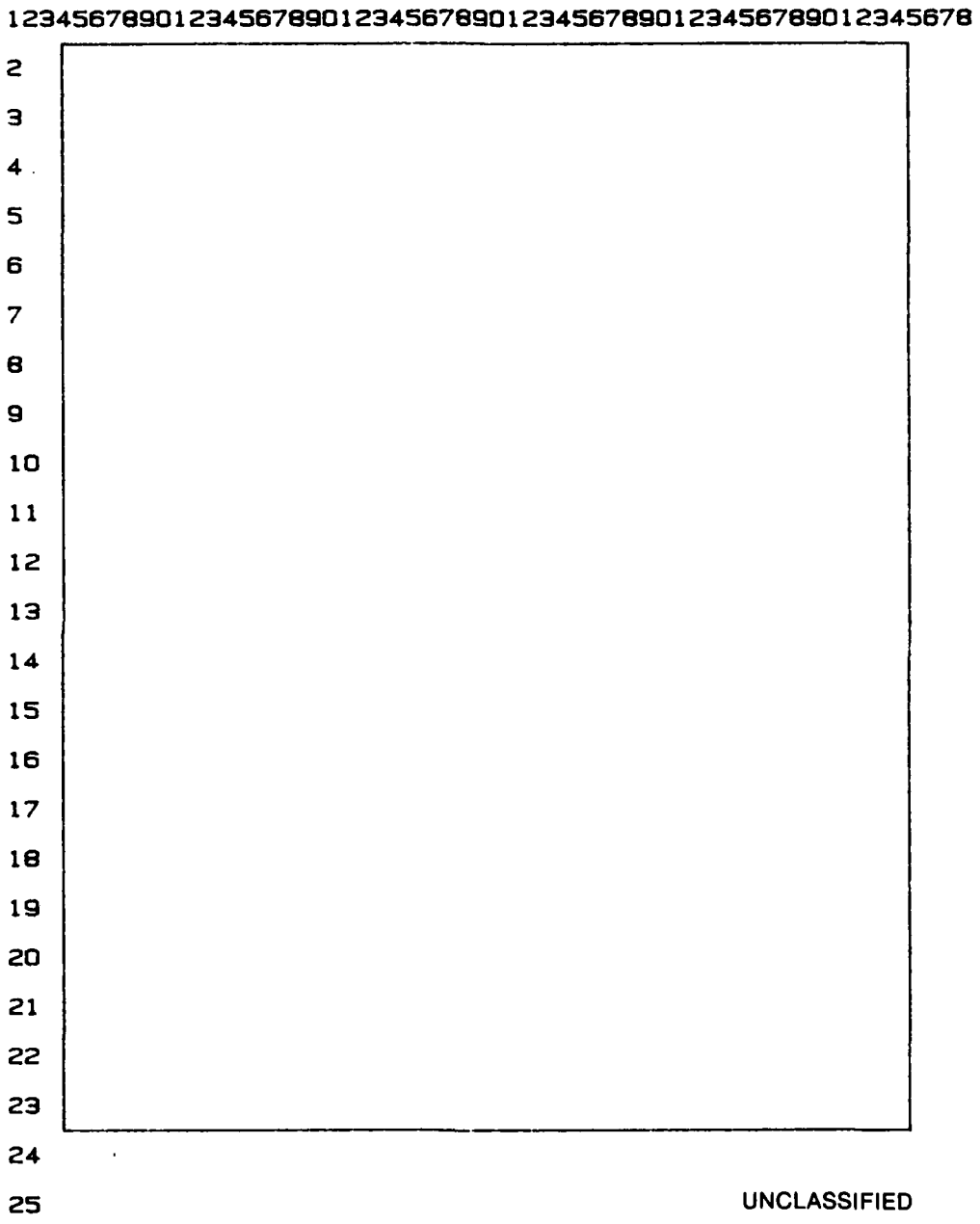

12345678901234567890123456789012345678901234567890123456789012345678
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
UNCLASSIFIED

Figure 16

INITIAL DISTRIBUTION

NAVAL UNDERWATER SYSTEMS CENTER
NEW LONDON, CT 06320
CODE 3421

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93940
CODE 62AB

HEADQUARTERS, U.S. ARMY
ESEIA/ADC-TP
FORT HUACHUCA, AZ 85613-5300

U.S. ARMY
CECOM/DRSEL-COM-RN-4
FORT MONMOUTH, NJ 07703

DEFENSE TECHNICAL INFORMATION CENTER
ALEXANDRIA, VA 22314

LAWRENCE LIVERMORE LABORATORY
PO BOX 808
LIVERMORE, CA 94550
L-156

END

FILMED

11-85

DTIC